普通高等教育“十一五”国家级规划教材

新编《信息、控制与系统》系列教材

系统仿真导论（第2版）

Introduction to System Simulation

肖田元 范文慧 编著

Xiao Tianyuan Fan Wenhui

清华大学出版社

作 者 简 介



肖田元 1947年生，清华大学教授，博士生导师，获国务院颁发的政府特殊津贴。现任清华大学自动化系学术委员会主任，国家CIMS工程技术研究中心常务副主任，中国系统仿真学会副理事长，中国自动化学会系统仿真专业委员会主任委员，《系统仿真学报》编委会常务副主任。主要研究领域是：系统仿真与虚拟制造、CIMS与网络化制造。作为项目负责人和主要参加者，承担863、国家自然科学基金、北京市科委以及企业的研究、攻关及工程应用项目近30余项。发表论文200余篇，SCI检索10余篇，EI检索50余篇，出版著作9部。获国家科技进步二等奖1项，三等奖2项，部级一等奖3项，二等奖2项。



范文慧 1966年生，博士，副教授，清华大学自动化系系统集成研究所副所长，中国系统仿真学会常务理事，中国自动化学会系统仿真专业委员会副秘书长，《系统仿真学报》编委。近年来一直从事仿真与虚拟制造的研究、开发与应用工作。在国内外核心期刊和会议上发表论文60余篇，作为第一负责人完成和实施国家自然科学基金2项、国家863重大项目3项、国家863基础研究项目1项，企业应用项目5项，获得北京市科学技术进步奖二等奖1项、三等奖1项和教育部科学技术进步二等奖1项。

ISBN 978-7-302-20445-9
9 787302 204459 >
定价：39.80元

普通高等教育“十一五”国家级规划教材

新编《信息、控制与系统》系列教材

系统仿真导论（第2版）

Introduction to System Simulation

肖田元 范文慧 编著
Xiao Tianyuan Fan Wenhui

清华大学出版社
北京

内 容 简 介

仿真是研究、设计、分析各种复杂系统的重要工具，已经成为与理论研究、实验研究并行的人类认识世界的重要方法。

作为普通高等教育“十一五”国家级规划教材，本书根据8年来的教学实践和仿真技术近年来的发展，在第一版(2000年)基础上进行了修订与扩展，其目的是为相关专业的本科高年级学生提供有关仿真学科与技术更全面、更系统、更新颖的教材。

全书共12章，第1章是概论，其余11章分为三篇，即连续系统仿真篇5章(第2～第6章)、离散事件系统仿真篇4章(第7～第10章)、现代仿真技术篇2章(第11、第12章)。几乎每章后面均附有习题，书后还附有实验指示书，包括进行仿真实验的源程序。

本书可作为自动化、机械工程、经济管理、系统工程、工业工程等专业大学本科高年级学生仿真课程的教材，其中带“*”的章节是选修内容；也可作为研究生的参考教材。对于工程技术人员，本书可作为应用仿真技术的参考书。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目(CIP)数据

系统仿真导论/肖田元，范文慧编著. —2版. —北京：清华大学出版社，2010.2
(新编《信息、控制与系统》系列教材)
ISBN 978-7-302-20445-9

Ⅰ. ①系… Ⅱ. ①肖… ②范… Ⅲ. ①系统仿真—高等学校—教材 Ⅳ. ①TP391.9

中国版本图书馆CIP数据核字(2009)第106476号

责任编辑：王一玲
责任校对：白　蕾
责任印制：何　芊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦A座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：30　**字　数**：726千字
版　　次：2010年2月第2版　　**印　　次**：2010年2月第1次印刷
印　　数：1～3000
定　　价：39.80元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010)62770177转3103　　产品编号：017537-01

新编《信息、控制与系统》系列教材
出 版 说 明

信息、控制与系统学科是在20世纪上半叶形成和发展起来的一门新兴技术科学。在人类探索自然和实现现代化的进程中，信息、控制与系统学科的理论、方法和技术始终起着重要的和基础的作用。基于信息、控制与系统科学的自动化的发展和应用水平在一定意义上是一个国家和社会的现代化程度的重要标志之一。本系列教材是关于信息、控制与系统学科所属各个领域的基本理论和前沿技术的一套高等学校系列教材。

本系列教材所涉及的范围包括信号和信息处理、模式识别、知识工程、控制理论、智能控制、过程和运动控制、传感技术、系统工程、机器人控制、工业自动化、计算机控制和仿真、网络化系统、电子技术等方面。主要读者对象为自动控制、工业自动化、计算机科学和技术、电气工程、机械工程、化工工程和热能工程等专业有关的高年级大学生和研究生，以及工作于相应领域和部门的科学工作者和工程技术人员。

10多年前，清华大学出版社会同清华大学自动化系，曾经组编出版过一套《信息、控制与系统》系列教材，产生了较大的社会影响，其中多数著作获得过包括国家级教学成果奖和部委优秀教材奖在内的各种奖励，至今仍为国内众多院校所采用，并被广大相关领域科技人员作为进修和自学读物。我们现在组编的这套新编《信息、控制与系统》系列教材，从一定意义上说，就是先前那套教材的延伸和发展，以反映近年来学科的发展和在科学研究与教学实践上的新成果和新进展，以适应当前科技发展和教学改革的新形势和新需要。列入这套新编系列教材中的著作，大多是清华大学自动化系开设的课程中经过较长教学实践而形成的，既有多年教学经验和教学改革基础上的新编著的教材，也有部分原系列教材的更新和修订版本。这套新编系列教材总体上仍将保持原系列教材求新与求实的风格，力求反映所属领域的基本理论和新近进展，力求做到学科先进性和教学适用性统一。需要说明的是，此前我们曾以《信息技术丛书》为名组编这套教材，并已出版了若干种著作。现为使“书”和“名”更为相符，这些已出版的著作将在重印或再版时列入这套新编系列教材。

我们希望，这套新编系列教材，既能为在校大学生和研究生的学习提供内容先进、论述系统和教学适用的教材或参考书，也能为广大科学工作者与工程技术人员的知识更新与继续学习提供适合的和有价值的进修或自学读物。我们同时要感谢使用本系列教材的广大教师、学生和科技工作者的热情支持，并热忱欢迎提出批评和意见。

新编《信息、控制与系统》系列教材编委会

2002年6月

新编《信息、控制与系统》系列教材编委会

顾　　问　李衍达　吴　澄　边肇祺　王桂增

主　　编　郑大钟

编　　委　徐文立　王　雄　萧德云　杨士元　肖田元

张贤达　周东华　钟宜生　张长水　王书宁

范玉顺　蔡鸿程

责任编辑　王一玲

前　言

仿真科学与技术是以建模与仿真理论为基础，建立并利用模型，以计算机系统、物理效应设备及(/或)仿真器为工具，对研究对象进行分析、设计、运行和评估的一门综合性、交叉性学科。

仿真科学与技术已经成为与理论研究、实验研究并行的人类认识世界的重要方法，在关系国家实力和安全的关键领域，如航空航天、信息、生物、材料、能源、先进制造、农业、教育、军事、交通、医学等领域，发挥着不可或缺的作用。

经过近一个世纪的发展历程，“仿真科学与技术”已形成独立的知识体系，包括：由仿真建模理论、仿真系统理论和仿真应用理论构成的理论体系；由自然科学的公共基础知识，各应用领域内的基础专业知识和仿真科学与技术的基础专业知识综合而构成的知识基础；由基于相似理论的仿真建模，基于整体论的网络化、智能化、协同化、普适化的仿真系统构建和全系统、全寿命周期、全方位的仿真应用思想综合而成的方法论。

在2000年第一版的基础上，本书根据仿真科学与技术的最新发展和8年来的教学实践，进行了内容修改、扩充以及结构调整，力图更全面、更新颖、更系统地向读者介绍仿真科学与技术的知识。全书共12章，第1章是概论，其余11章分为三篇，即连续系统仿真篇5章(第2～第6章)、离散事件系统仿真篇4章(第7～10章)、现代仿真技术篇2章(第11、第12章)。

本书第1章概论，使初次接触仿真的读者对仿真科学与技术有一个概念性了解。新版除了保留原来的基本论述外，增加了对仿真科学与技术的基础理论——相似理论的介绍，也对当前仿真技术研究的热点进行了讨论。

连续系统仿真篇分为5章。第2章对连续系统模型设计进行讨论，新版增加了计算机辅助模型设计内容，并以电路系统、多体系统为对象讨论了模型设计方法；第3章介绍经典的连续系统仿真建模方法学，除了讨论经典的数值积分的单步法和多步法之外，还讨论了病态系统仿真建模方法；第4章讨论了现代的连续系统仿真建模技术——离散相似法，分别按时域和频域进行论述。到此，对有关集中参数的连续系统仿真技术作了比较全面的介绍，

新版增加了一个案例讨论，以便读者进一步理解如何开展实际系统的仿真研究。考虑到MATLAB应用的广泛性，新版的第5章专门安排了有关连续系统仿真软件的内容，特别是对MATLAB的模型设计方法、仿真实现技术进行了介绍。第6章提供有关分布参数系统的仿真建模技术，除了传统的差分法之外，考虑到有限元法已经得到广泛应用，因此新版对有限元法的原理也进行了介绍。

离散事件系统仿真篇分为4章。第7章介绍离散事件系统仿真的基本概念，通过两类常见的离散事件系统——排队系统和库存系统的仿真使读者对该类仿真有一个总体认识。第8章讨论离散事件系统建模方法学。随机性是这类系统的重要特征，讨论了如何从观测数据确定随机变量模型以及各类典型的连续与离散随机变量的产生等。第9章讨论了离散事件系统仿真建模方法学，对事件调度法、活动扫描、进程交互法以及三阶段法系统地进行了讨论，并进行了比较；在此基础上，新版增加了对离散事件系统仿真软件的发展历史以及典型的仿真软件的介绍。第10章对离散事件系统仿真输出分析进行了比较详细的讨论，包括单系统、多系统仿真运行与输出分析技术等。

本书的第3篇是介绍现代仿真技术。在第11章“并行仿真与分布式仿真”中，除了保留原版有关内容外，新版增加了并行仿真的实现技术以及高层体系结构(HLA)的相关内容，并对其最新的研究进展进行了介绍。第12章介绍了“虚拟现实技术”，新版增加了当前典型的开发虚拟现实的工具的介绍。

本书作为理工类高等学校学生仿真课程的教材，几乎每章后面均附有习题，考虑到系统仿真是一门实验性很强的学科，书的最后附有实验指示书，包括进行仿真实验的源程序。程序是用C语言编写的。

本书的参考教学学时为48～60学时。学习本课程之前，学生必须已修过“自动控制理论”、“概率论与数理统计”、“计算机程序设计语言”等课程。

本书可作为自动化、机械工程、经济管理、系统工程、工业工程等专业大学高年级学生的仿真课程教材，其中带“*”的章节是选修内容；也可作为研究生的参考教材。

本书是教育部批准的普通高等教育“十一五”国家级规划教材，由清华大学自动化系肖田元、范文慧共同编写，全书由肖田元审定。

由于作者的水平有限，恳请读者对书中的缺点和错误给予批评指正。

作　者

2009年8月

目　录

第 1 章　概论 …… 1

1.1　系统、模型与仿真 …… 1

1.1.1　系统 …… 1

1.1.2　模型 …… 2

1.1.3　仿真 …… 4

1.1.4　仿真科学与技术的发展沿革 …… 6

1.2　系统仿真的一般知识 …… 7

1.2.1　相似理论 …… 7

1.2.2　基于相似理论的系统仿真 …… 8

1.2.3　系统仿真的类型 …… 10

1.2.4　系统仿真的一般步骤 …… 13

1.3　仿真科学与技术的应用 …… 13

1.3.1　仿真在系统设计中的应用 …… 14

1.3.2　仿真在系统分析中的应用 …… 14

1.3.3　仿真在教育与训练中的应用 …… 15

1.3.4　仿真在产品开发及制造过程中的应用 …… 15

1.4　当前仿真科学与技术研究的热点 …… 16

1.4.1　网络化仿真技术 …… 16

1.4.2　复杂系统/开放复杂巨系统的建模与仿真 …… 17

1.4.3　综合自然环境的建模与仿真 …… 20

1.4.4　基于普适计算技术的普适仿真技术 …… 21

1.4.5　基于高性能计算机的仿真 …… 22

1.5　小结 …… 22

习题 …… 23

参考文献 …… 23

第一篇 连续系统仿真篇

第 2 章 连续系统模型 …… 27
2.1 连续系统模型描述 …… 27
2.1.1 连续时间模型 …… 27
2.1.2 离散时间模型 …… 29
2.1.3 连续-离散混合模型 …… 31
2.2 连续系统模型设计 …… 32
2.2.1 外部模型到内部模型的变换 …… 32
2.2.2 系统状态初始值变换 …… 36
2.2.3 面向结构图的模型变换 …… 38
2.3 计算机辅助模型设计 …… 44
2.3.1 基于计算机的建模 …… 44
2.3.2 解释结构建模 …… 46
2.4 电路系统模型设计 …… 49
2.4.1 电路系统模型 …… 49
2.4.2 电路系统的拓扑 …… 51
2.4.3 改进节点法 …… 54
2.4.4 小结 …… 56
2.5 多体系统模型设计* …… 57
2.5.1 多体系统建模基本概念 …… 57
2.5.2 多体系统拓扑构型 …… 58
2.5.3 基于拉格朗日坐标的多刚体系统运动学模型描述 …… 61
2.5.4 基于 Haug 笛卡尔方法的多刚体系统运动学模型描述 …… 65
2.5.5 多刚体系统动力学模型描述 …… 68
2.5.6 小结 …… 70
习题 …… 70
参考文献 …… 73

第 3 章 经典的连续系统仿真建模方法学 …… 74
3.1 连续系统离散化原理 …… 74
3.2 龙格-库塔法 …… 76
3.2.1 龙格-库塔法基本原理 …… 76
3.2.2 龙格-库塔法的误差估计及步长控制 …… 80
3.2.3 实时龙格-库塔法 …… 84
3.3 线性多步法 …… 85
3.3.1 线性多步法基本原理 …… 86
3.3.2 线性多步法误差分析 …… 90

3.3.3 关于变步长的讨论 …… 92
3.4 稳定性分析 …… 94
3.5 病态系统仿真* …… 96
3.5.1 病态系统的定义 …… 96
3.5.2 线性病态系统仿真 …… 97
3.5.3 吉尔(Gear)法 …… 99
3.5.4 单步多值法 …… 101
3.5.5 病态探测与误差估计 …… 105
3.5.6 半隐式龙格-库塔法 …… 111
3.5.7 多帧速算法 …… 112
3.6 间断特性的仿真建模* …… 115
3.6.1 条件函数零点搜索法 …… 116
3.6.2 平均值法 …… 118
3.6.3 间断点估计法 …… 122
3.7 小结 …… 123
习题 …… 124
参考文献 …… 125

第4章 离散相似法 …… 127
4.1 时域离散相似法 …… 127
4.1.1 基本原理 …… 127
4.1.2 状态转移矩阵的计算 …… 131
4.1.3 增广矩阵法 …… 134
4.2 面向结构图的非线性系统仿真 …… 136
4.2.1 典型线性动态环节$\boldsymbol{\Phi}(T)$,$\boldsymbol{\Phi}_m(T)$,$\hat{\boldsymbol{\Phi}}_m(T)$的计算 …… 137
4.2.2 常用非线性环节计算子程序 …… 139
4.2.3 非线性系统离散相似法仿真程序 CSS02.C …… 140
4.3 频域离散相似法 …… 144
4.3.1 替换法 …… 144
4.3.2 根匹配法 …… 148
4.3.3 可调整积分法 …… 151
4.4 离散相似法应用举例 …… 158
4.4.1 机械谐振分析 …… 158
4.4.2 模型设计 …… 160
4.4.3 仿真设计 …… 167
4.4.4 仿真结果分析 …… 178
4.4.5 小结 …… 180
4.5 采样控制系统仿真 …… 181
4.5.1 采样控制系统 …… 181

4.5.2 采样周期与仿真步距 …… 182
4.5.3 纯延迟环节的仿真模型 …… 186
4.5.4 采样控制系统仿真举例 …… 187
4.6 小结 …… 189
习题 …… 190
参考文献 …… 191

第5章 集中参数连续系统仿真语言 …… 192
5.1 发展沿革 …… 192
5.2 MATLAB简介 …… 194
5.2.1 MATLAB发展历程 …… 194
5.2.2 MATLAB系统构成 …… 195
5.2.3 MATLAB工具箱 …… 195
5.2.4 MATLAB 7.0/Simulink 6.0特点 …… 196
5.3 MATLAB的模型设计 …… 199
5.3.1 线性连续系统Simulink描述 …… 199
5.3.2 线性离散时间系统Simulink描述 …… 200
5.3.3 连续-离散时间混合系统Simulink描述 …… 200
5.4 MATLAB仿真建模技术 …… 203
5.4.1 连续系统仿真建模 …… 203
5.4.2 离散时间系统与连续-离散时间混合系统仿真建模 …… 206
5.4.3 间断点及代数环仿真建模 …… 206
5.5 MATLAB模块化集成技术 …… 207
5.5.1 数据驱动模型 …… 207
5.5.2 子系统集成 …… 208
5.5.3 MATLAB典型子系统 …… 208
5.6 小结 …… 211
习题 …… 211
参考文献 …… 212

第6章 分布参数系统仿真建模* …… 213
6.1 概述 …… 213
6.2 差分法原理 …… 214
6.2.1 对流方程差分法 …… 215
6.2.2 差分格式的相容性、收敛性及稳定性 …… 216
6.3 典型分布参数系统的差分法 …… 221
6.3.1 扩散方程的菱形法及跳点法 …… 221
6.3.2 对流方程的耗散中心差公式 …… 223
6.3.3 椭圆方程的差分法 …… 224

6.4 线上求解法 …… 226
6.5 有限元方法预备知识 …… 228
6.5.1 微分方程的等效积分形式与加权余量法 …… 228
6.5.2 弹性力学的基本方程 …… 232
6.6 有限元方法原理 …… 237
6.6.1 三角形单元的有限元格式 …… 237
6.6.2 基于最小位能原理的有限元方程 …… 240
6.6.3 广义坐标有限元法的一般格式 …… 243
6.7 小结 …… 246
习题 …… 247
参考文献 …… 248

第二篇 离散事件系统仿真篇

第 7 章 离散事件系统仿真基础 …… 251
7.1 基本概念 …… 251
7.2 仿真钟的推进 …… 253
7.3 单服务台排队系统仿真 …… 256
7.3.1 排队系统的基本概念 …… 256
7.3.2 单服务台排队系统仿真举例 …… 257
7.4 库存系统的仿真 …… 261
7.4.1 库存系统的基本概念 …… 261
7.4.2 确定性库存系统 …… 262
7.4.3 随机库存系统 …… 263
7.4.4 随机库存系统仿真举例 …… 264
7.5 离散事件系统仿真研究的一般步骤 …… 267
7.6 小结 …… 269
习题 …… 269
参考文献 …… 270

第 8 章 随机变量模型的确定与产生 …… 271
8.1 引言 …… 271
8.2 随机变量模型的确定 …… 271
8.2.1 分布参数的确定 …… 272
8.2.2 分布类型的假设 …… 276
8.2.3 实验分布 …… 283
8.3 拟合优良度检验 …… 284
8.3.1 χ^2 检验 …… 284
8.3.2 柯尔莫哥洛夫-斯米尔洛夫(K-S 检验) …… 286

8.4 随机数发生器 …… 288
8.4.1 线性同余发生器 …… 288
8.4.2 组合发生器 …… 290
8.5 随机数发生器的测试 …… 291
8.5.1 均匀性检验 …… 291
8.5.2 独立性检验 …… 291
8.6 随机变量产生的原理 …… 293
8.6.1 反变换法 …… 294
8.6.2 组合法 …… 296
8.6.3 卷积法 …… 297
8.6.4 舍选法 …… 298
8.7 典型随机变量的产生 …… 300
8.7.1 连续随机变量的产生 …… 300
8.7.2 离散随机变量的产生 …… 304
8.8 小结 …… 307
习题 …… 307
参考文献 …… 309

第 9 章 离散事件系统仿真策略与仿真语言 …… 310
9.1 离散事件系统仿真策略 …… 310
9.1.1 主要术语 …… 310
9.1.2 事件调度法(event scheduling) …… 311
9.1.3 活动扫描法(activity scanning) …… 313
9.1.4 进程交互法(process interactive) …… 314
9.1.5 三阶段法 …… 316
9.1.6 四种仿真策略的比较 …… 319
9.2 离散事件系统仿真语言 …… 320
9.2.1 离散事件系统仿真软件发展历程 …… 320
9.2.2 典型仿真语言简介 …… 322
9.3 小结 …… 329
习题 …… 330
参考文献 …… 331

第 10 章 离散事件系统仿真输出分析 …… 332
10.1 仿真输出分析方法的分类 …… 332
10.2 终止型仿真输出分析 …… 334
10.2.1 固定样本长度法 …… 334
10.2.2 终止型序贯程序法 …… 335
10.3 稳态型仿真输出分析 …… 336

10.3.1 批均值法 …… 336
10.3.2 稳态型序贯法 …… 338
10.3.3 重新产生法 …… 341
10.3.4 重复删除法 …… 343
10.4 系统性能比较 …… 346
10.4.1 两系统性能比较 …… 346
10.4.2 多系统性能比较 …… 347
10.5 仿真结果分析应用举例 …… 349
10.6 方差减少技术* …… 352
10.6.1 公共随机数法 …… 353
10.6.2 对偶变量法 …… 356
10.6.3 控制变量法 …… 357
10.7 小结 …… 358
习题 …… 358
参考文献 …… 359

第三篇 现代仿真篇

第 11 章 并行仿真与分布仿真* …… 363
11.1 前言 …… 363
11.2 并行仿真 …… 363
11.2.1 并行仿真的基本概念 …… 363
11.2.2 并行处理原理及并行仿真算法 …… 367
11.2.3 面向微分方程的并行仿真 …… 370
11.2.4 并行仿真任务划分 …… 373
11.2.5 并行仿真的实现 …… 375
11.3 分布交互式仿真 …… 378
11.3.1 分布交互式仿真的发展 …… 378
11.3.2 分布交互式仿真基本原理 …… 380
11.4 基于高层体系结构的仿真 …… 384
11.4.1 HLA 体系结构 …… 384
11.4.2 HLA 标准 …… 386
11.4.3 HLA 运行支撑环境(RTI) …… 391
11.4.4 小结 …… 394
11.5 并行仿真与分布仿真的进展 …… 395
11.5.1 可扩展建模与仿真框架(XMSF) …… 395
11.5.2 面向服务框架(SOA) …… 396
11.5.3 仿真网格(SimGrid) …… 397
11.5.4 小结 …… 401
参考文献 …… 402

第 12 章 虚拟现实技术 …… 403
12.1 虚拟现实的特征及基本构成 …… 403
12.2 虚拟现实技术的发展概况 …… 407
12.2.1 虚拟现实技术的发展历程 …… 407
12.2.2 虚拟现实技术发展的瓶颈 …… 408
12.2.3 当前流行的虚拟现实系统 …… 409
12.3 位置跟踪通道 …… 411
12.3.1 三维电磁传感器 …… 411
12.3.2 三维超声波传感器 …… 412
12.3.3 数据手套(data glove) …… 413
12.4 视觉通道 …… 414
12.4.1 视景的生成过程 …… 414
12.4.2 基于投影变换原理三维立体图的生成算法 …… 414
12.4.3 典型的立体视觉设备 …… 417
12.5 听觉通道 …… 418
12.5.1 虚拟声音的产生 …… 418
12.5.2 虚拟声音的定位 …… 420
12.6 触觉、力反馈 …… 421
12.6.1 人体触觉系统 …… 422
12.6.2 触觉/力反馈技术 …… 423
12.7 虚拟场景的生成 …… 427
12.7.1 虚拟场景静态模型的建立 …… 427
12.7.2 虚拟场景动态模型的建立 …… 429
12.8 典型的虚拟现实系统软件 …… 430
12.9 小结 …… 434
参考文献 …… 434

附录1 实验指示书及仿真源程序 …… 436
实验一 面向方程的数值积分方法仿真 …… 436
实验二 面向结构图的离散相似法仿真 …… 442
实验三 随机数产生及均匀性与独立性测试 …… 450
实验四 仿真输出数据分析 …… 453

附录 2 英文缩写词索引 …… 459

第1章 概　论

“仿真”对应的英文单词是“Simulation”，也可译为模拟，为了与模拟计算机“Analog Computer”中的“Analog”以示区别，在规范术语和译名时，1979年我国专家和学者们建议将“Simulation”译为“仿真”，并一直沿用至今。

仿真已成功应用于工业、农业、商业、教育、军事、交通、社会、经济、医学、生命、生活服务等众多领域，并在航空、航天、信息、生物、材料、能源、先进制造等高新技术领域中成为系统论证、设计、实验、分析、维护、辅助决策及人员培训的重要手段与工具。实践表明，仿真已经成为认识和改造客观世界的重要方法。

1.1 系统、模型与仿真

1.1.1 系统

“系统”对应的英语单词是system，它源自希腊语 systèma。正如G. 戈登在其所著的《系统仿真》一书中所说：“系统这个术语已经在各个领域用得如此广泛，以至于很难给它下一个定义。”系统这一词最早出现于古希腊原子论创始人德谟克利特（公元前460—公元前370年）的著作《世界大系统》。该书明确地论述了关于系统的含义：“任何事物都是在关联中显现出来的，都是在系统中存在的，系统关联规定每一事物，而每一关联又能反映系统关联的总貌。”戈登在总结前人思想的基础上，将系统定义为“按照某些规律结合起来，互相作用、互相依存的所有实体的集合或总和”。美国传统词典（双解）对系统的定义是：组成一个复杂的整体的一组互相作用、互相关联或互相依存的元素（system: A group of interacting, interrelated, or interdependent elements forming a complex whole）。

根据这些定义，我们可以将一个理发馆定义为一个系统。服务员和顾客是该系统中的实体。顾客按某种规律到达，服务员根据顾客的要求，按一定的程序为其服务，服务完毕后

顾客离去。在该系统中，顾客和服务员互相作用，顾客到达模式影响着服务员的工作忙闲状态和理发馆的排队状态，而服务员的多少和服务效率高低也影响着顾客接受服务的质量。

我们也可以把实现电动机调速的各个实体定义为一个系统，称为电动机调速系统，该系统包括电动机、测速元件、比较元件以及控制器等。它们相互作用以实现按给定要求调节电动机的速度。

在定义一个系统时，首先要确定系统的边界。尽管世界上的事物是相互联系的，但当我们研究某一对象时，总是要将该对象与其环境区别开来。边界确定了系统的范围，边界以外对系统的作用称为系统的输入，系统对边界以外的环境的作用称为系统的输出。

尽管世界上的系统千差万别，但人们总结出描述系统“三要素”，即实体、属性、活动。实体确定了系统的构成，也就确定了系统的边界；属性也称为描述变量，描述了每一实体的特征，其中系统的状态对描述系统在任意时刻、相对于研究对象来说是必需的；活动定义了系统内部实体之间的相互作用，从而确定了系统内部发生变化的过程。

系统可以被划分为离散的或连续的两类。“连续系统”是指状态变量随时间连续改变的系统，“离散系统”是指其状态变量只在某个离散时间点集合上发生变化的系统。实际上很少有系统是完全离散的或完全连续的，但对于大多数系统来说，由于某一类型的变化占据主导地位，因此会有可能将系统划分为离散的或连续的。

1.1.2 模型

在一般意义上，模型是一种替代(proxy)，用于代表其对象以便得到更好的定义。从应用的角度，模型不是原对象的复制，而是根据不同的使用目的，选取原对象的若干方面进行本质的抽象和简化。模型有多种形式，典型的有：

(1) 实物模型(比例模型、模拟模型或原型)，例如用于新飞机设计的风洞实验的机翼模型；

(2) 方程式或逻辑表达式表示的系统(数学模型)，如为预测化学反应的最终产品的质量或能量的平衡方程，预测飞机飞行的计算机程序等；

(3) 图(图标模型)，如用于记录地理数据的地图，或用于设计机器零部件的几何模型；

(4) 智力模型，如为指导人的行为而建立的人与环境的关系模型；

(5) 语言或文字描述(语言模型)，如为指导生物实验或医生手术的方案等。

总之，按照系统论的观点，模型是对真实系统的描述、模仿或抽象，即将真实系统的本质用适当的表现形式(如文字、符号、图表、实物、数学公式等)加以描述。

开发模型的目的是用模型作为替代来帮助人们对原物进行假设、定义、探究、理解、预测、设计，或者与原物某一部分进行通信。为了研究、分析、设计和实现一个系统，需要进行试验。试验的方法基本上可分为两大类：一种是直接在真实系统上进行，另一种是先构造模型，通过对模型的试验来代替或部分代替对真实系统的试验。历史上大多采用在实际系统上做试验，随着科学技术的发展，尽管第一种方法在某些情况下仍然是必不可少的，但第二种方法日益成为人们更为常用的方法。在模型上进行试验日益为人们所青睐，主要原因在于：

(1) 系统还处于设计阶段，真实的系统尚未建立，人们需要更准确地了解未来系统的性能，这只能通过对模型的试验来了解；

(2) 在真实系统上进行试验可能会引起系统破坏或发生故障，例如，对一个处于运行状态的化工系统或电力系统进行没有把握的试验将会冒巨大的风险；

(3) 需要进行多次试验时，难以保证每次试验的条件相同，因而无法准确判断试验结果的优劣；

(4) 试验后，系统难以复原；

(5) 试验时间太长或费用昂贵。

通过模型试验可以很好地解决这些问题。与通过实物系统试验相比，模型通常更易于理解，模型结构的变化更容易实现，模型的行为特性更易于掌握。因此，在模型上进行试验已经成为人们科学研究与工程实践中不可缺少的手段。

形式化模型是所有科学与技术学科所采用的主要描述形式，社会与管理学科也越来越多地采用形式化模型。一个系统的模型可以定义为如下集合结构

$$S=(T,X,\Omega,Q,Y,\delta,\lambda)$$

其中

T——时间基，描述系统变化的时间坐标，T 为整数则称为离散时间系统，T 为实数则称为连续时间系统。

X——输入集，代表外部环境对系统的作用。通常 X 被定义为 R^n，其中 $n\in I^+$，即 X 代表 n 个实值的输入变量。

Ω——输入段集，描述某个时间间隔内的输入模式，是(X,T)的一个子集。

Q——内部状态集，是系统内部结构建模的核心。

δ——状态转移函数，定义系统内部状态是如何变化的。它是一个映射：$\delta: Q\times\Omega\to Q$，其含义是：若系统在 t_0 时刻处于状态 q，并施加一个输入段 ω：$<t_0,t_1>\to X$，则 $\delta(q,\omega)$ 表示系统处于 t_1 状态。

Y——输出段集，系统通过它作用于环境。

λ——输出函数，它是映射：$\lambda: Q\times X\times T\to Y$。输出函数给出了一个输出段集。

上面给出了系统模型的一般描述。在实际建模时，由于要求不同，模型描述的详细程度也不尽相同，亦称为表示的层次不同，具体的有：

(1) 行为层次　亦称为输入输出层次。该层次的模型将系统视为一个“黑盒”，在输入信号的作用下，只对系统的输出进行测量。

(2) 分解结构层次　将系统看成若干个黑盒连接起来，定义每个“黑盒”的输入与输出，以及它们相互之间的连接关系。

(3) 状态结构层次　不仅定义了系统的输入与输出，而且还定义了系统内部的状态集及状态转移函数。

上面我们从系统的定义出发给出了集合结构的模型描述形式，显然，这种描述太一般化了，难以在实际中加以采用。人们在长期的研究与应用中，创造出了适用于不同对象研究分析要求的模型描述形式，Orën 进行了总结，将模型形式加以分类如表 1.1 所示。

表 1.1 模型分类

模型描述变量的轨迹	模型的时间集合	模型形式	变量范围	
			连续	离散
空间连续变化模型	连续时间模型	偏微分方程	√	
空间不连续变化模型		常微分方程	√	
离散(变化)模型	离散时间模型	差分方程	√	√
		有限状态机		√
		马尔可夫链		√
	连续时间模型	活动扫描	√	√
		事件调度	√	√
		进程交互	√	√

建立一个实际系统的模型一般包括两方面的内容：第一是建立模型结构，包括确定系统边界，鉴别系统的实体、属性和活动，基于其内部规律的理解给出其描述形式。第二是根据研究目标提供数据，即提供所研究的活动中产生的有效数据。

一般说来，模型结构应具有以下性质：

(1) 相似性：模型与被研究对象在属性上具有相似的特性和变化规律。模型作为“替身”，应与实际对象“原型”在本质上是相似的。

(2) 简单性：模型不是越复杂越好，相反，在满足相似性的前提下，模型应当尽量简单。因此，要根据研究的目标，忽略实际系统中的一些次要因素非常重要。

(3) 多面性：由于实际对象的复杂性，人们研究的目的往往是不完全相同的，因而对系统的理解、所搜集的数据也不完全相同，从而得到的模型结构也不唯一。因此，模型如何满足多方面、多层次研究的需求，这是建模时需要特别加以考虑的。

模型的有效性是建模的基本要求，它反映了模型表示实际对象的充分程度，一般是用实际系统的观测行为及数据与模型产生的行为与数据之间的吻合程度来度量。它可分为三个级别：

(1) 复制有效(replicatively valid)：建模者将实际对象视为一个黑箱(black box)，仅在输入输出行为层次上认识系统，即只要求模型产生的输入输出数据与从实际对象得到的输入输出相匹配。

(2) 预测有效(predictively valid)：建模者对实际对象的总体结构及其内部规律有一定了解，但对其详细结构仍然缺乏足够的信息和数据，所建模型不但与已有的实际对象的行为和数据相匹配，而且可以用来预测实际对象未知的行为和信息。

(3) 结构有效(structurally valid)：建模者对实际对象的详细结构有清楚理解，即系统的子系统的构成、子系统的内部结构、子系统之间的相互作用等均相匹配。这样，模型不但能用于观测实际系统的行为，而且能反映实际系统产生行为的操作过程。

1.1.3 仿真

按照牛津英语词典的定义，仿真是用相似的模型、环境和设备模仿某个环境或系统(经济的、机械的等)的行为的技术，或者是为更方便地获取信息，或者是为培训人(The

technique of imitating the behavior of some situation or system (Economic, Mechanical etc.) by means of an analogous model, situation, or apparatus, either to gain information more conveniently or to train personnel.)；仿真是一组广泛的方法，用于研究和分析实际的或理论的系统的行为和性能(A broad collection of methods, used to study and analyze the behavior and performance of actual or theoretical systems.)。

1961 年，G. W. Morgenthater 首次对“仿真”进行了技术性定义，即“仿真意指在实际系统尚不存在的情况下对于系统或活动本质的实现”。另一个典型的对“仿真”进行技术性定义的是 Korn。他在 1978 年的著作《连续系统仿真》中将仿真定义为“用能代表所研究的系统的模型作实验”。1982 年，Spriet 进一步将仿真的内涵加以扩充，定义为“所有支持模型建立与模型分析的活动即为仿真活动”。Orën 在 1984 年在给出了仿真的基本概念框架“建模—实验—分析”的基础上，提出了“仿真是一种基于模型的活动”的定义，被认为是现代仿真技术的一个重要概念。实际上，随着科学技术的进步，特别是信息技术的迅速发展，“仿真”的技术含义不断地得以发展和完善，从 A. Alan 和 B. Pritsker 撰写的“仿真定义汇编”一文我们可以清楚地观察到这种演变过程。无论哪种定义，仿真基于模型这一基本观点是共同的。

综上所述，“系统、模型、仿真”三者之间有着密切的关系。系统是研究的对象，模型是系统的抽象，仿真是通过对模型的实验以达到研究系统的目的。

现代仿真技术大多是在计算机支持下进行的，因此，系统仿真也称为计算机仿真。计算机仿真是“干中学”原理的典型，即为了了解和学习系统，首先建立模型，然后运行模型。仿真活动犹如儿童通过玩来认识世界一样，儿童用玩具、图画来模仿如何与他人、动物以及环境交互。为了理解和认识客观世界的本来面目及其复杂性，人们必须建立人造对象并使其动态地运行，计算机的作用就是驱动人造对象和虚拟环境。计算机仿真有三个基本活动：设计实际的或设想的物理系统的模型(模型设计)、在计算机上执行模型(模型执行)、分析执行输出(模型分析)。联系这三个活动的是系统仿真的三要素，即系统、模型、计算机(包括硬件和软件)。它们的关系可用图 1.1 描述。

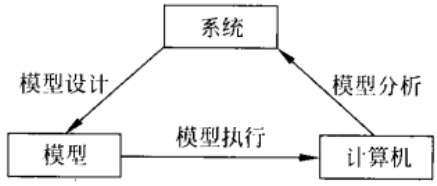


图 1.1 计算机仿真三要素及三个基本活动

传统上，“模型设计”这一活动属于系统辨识技术范畴，仿真技术则侧重在“仿真建模”，即针对不同形式的系统模型研究其求解算法，使其在计算机上得以实现。“模型执行”这一活动，也往往只注重“仿真程序”的检验(verification)；至于“模型分析”，即如何将仿真实验的结果与实际系统的行为进行比较这一根本的有效性问题(validation)还缺乏从方法学的高度进行研究。

现代仿真科学与技术的一个重要进展是将仿真活动扩展到上述三个方面，并将其统一到同一环境中。

在模型设计方面，除了传统的基于物理学、化学、生物学、社会学等基本定律及系统辨识等方法外，现代仿真技术提出了用仿真方法确定实际系统的模型。例如，根据某一系统在试验中所获得的输入输出数据，在计算机上进行仿真试验，确定模型的结构和参数；基于模型库的结构化建模，采用面向对象建模(object-oriented modeling)方法，在类库的基础上实现

模型拼合与重用；还有基于元模型(meta-model)的建模以及适应计算机软硬件环境的发展而不断研究和开发出的许多新方法等。

在模型执行方面，现代仿真技术采用模型与实验分离技术，即模型的数据驱动(data driven)。任何一个仿真问题可分为两部分：模型与实验，这一点，现代仿真技术与传统的仿真定义是一致的。其区别在于：现代仿真技术将模型又分为两部分参数模型和参数值，参数值属于实验框架的内容之一。这样，模型参数与其对应的参数模型分离开来。仿真实验时，只需对参数模型赋予具体参数值，就形成了一个特定的模型，从而大大提高了仿真的灵活性和运行效率。

在模型分析方面，现代仿真技术除了将实验框架与仿真运行控制区分开来外，一个实验框架定义一组条件，包括模型参数、输入变量、观测变量、初始条件、终止条件、输出说明。除此之外，与传统仿真区别在于，将输出函数的定义也与仿真模型分离开来。这样，当需要不同形式的输出时，不必重新修改仿真模型，甚至不必重新仿真运行，这样就能更方便地进行模型分析。

Orën 将上述思想加以总结，提出了现代仿真技术的概念框架(参见图 1.2)。在这个框架中，“仿真问题描述”对应于图 1.1 中的“模型设计”，其建模思想如前所述。“行为产生”对应于图 1.1 中的“模型执行”，只是将仿真输出独立于行为产生。而“模型行为及其处理”对应于“模型分析”。

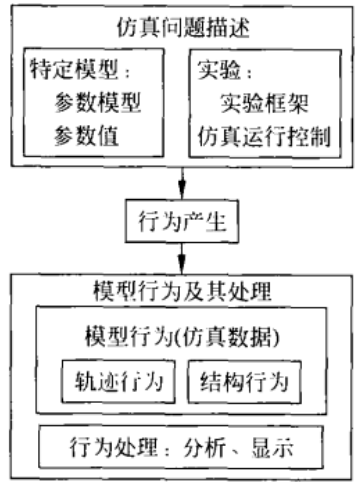


图 1.2 现代仿真的概念框架

1.1.4 仿真科学与技术的发展沿革

仿真科学与技术是工业社会由机械化向信息化时代前进中产生的新的技术学科。社会与经济发展需求的牵引和各门类科学与技术的发展，有力地推动了仿真科学与技术的发展。半个多世纪以来，仿真科学与技术在系统科学、控制科学、计算机科学等学科中孕育、交叉、综合和发展，并在各学科、各行业的实际应用中成长，逐渐突破孕育本学科的原学科范畴，成为一门新兴的学科。它的发展经历了如下几个阶段：

(1) 仿真技术的初级阶段

在第二次世界大战后期，火炮控制与飞行控制动力学系统的研究孕育了仿真科学与技术的发展。从 20 世纪 40 年代到 60 年代，相继研制成功了通用电子模拟计算机和混合模拟计算机。在导弹和宇宙飞船姿态及轨道动力学研究、阿波罗登月计划及核电站中仿真技术都得到应用。由于采用的工具是通用电子模拟计算机和混合模拟计算机，因而可称为模拟阶段。

(2) 仿真技术的发展阶段

20 世纪 70 年代，随着数字仿真机的诞生，仿真科学与技术不但在军事领域得到迅速发展，而且扩展到了许多工业领域，如培训飞行员的飞机训练仿真器、电站操作人员的仿真系统、汽车驾驶模拟器，以及复杂工业过程的仿真系统等，同时相继出现了一些从事仿真设备和仿真系统生产的专业化公司，使仿真科学与技术进入了数字仿真阶段。

(3) 仿真技术发展的成熟阶段

20 世纪 90 年代，被仿真的系统日益复杂，规模越来越大，在需求牵引和计算机科学与技术的推动下，为了更好地实现信息与仿真资源共享，促进仿真系统的互操作和重用，以美国为代表的发达国家在聚合级仿真、分布式交互仿真、先进的并行交互仿真的基础上，仿真科学与技术开始向高层体系结构(HLA)方向发展，实现多种类型仿真系统之间互操作、仿真模型组件重用的成熟阶段。

(4) 复杂系统仿真的新阶段

20 世纪末和本世纪初，对广泛领域的复杂性问题进行科学研究的需求进一步推动仿真技术的发展。仿真科学与技术在计算机技术、网络技术、图形图像技术、多媒体技术、软件工程、信息处理、控制论以及系统工程等技术的发展和支持下，逐渐发展形成了具有广泛应用领域的新兴的交叉学科——仿真科学与技术学科。

1.2 系统仿真的一般知识

1.2.1 相似理论

一般意义的相似理论是一种特定的科学观点，或者说它是一种科学的思维方式与方法，可以将它纳入认识论与方法论的范畴。在一切同类事物中相似现象是普遍的，而绝对相同几乎是不存在的，在一定条件下应用相似原则处理问题是可取的。

人们之所以能对实际系统进行仿真，是基于客观世界本身所固有的相似性以及人们对客观世界认识过程的相似性，这是系统仿真学科生存和发展的客观基础。

相似理论的基本原理包括同序结构原理、信息原理和支配原理等。这些原理反映了相似系统的形成和演变规律。系统仿真本质上就是依据相似规律人为地建立某种形式的相似模型去模拟实际系统，因此在仿真过程中应自觉地应用这些相似理论的基本原理。

(1) 同序结构原理

相似理论的同序结构原理认为，任何系统都有一定的序结构，序结构的规律性形成有序结构，其中包括空间有序、时间有序和功能有序。具体来说，空间有序表征系统组成要素的空间排列、组合和联系方式的规律性。时间有序表征系统要素随时间变化的运动规律。功能有序表征系统要素在相互作用过程中所表现出的各种功能发挥秩序的规律性。

系统的序结构决定了系统的整体特性。当系统序结构存在共同性时，系统之间存在相似性，其相似程度的大小取决于系统序结构的共同性程度的大小。基于系统相似性的仿真模型应以某种形式、在某种程度上反映实际系统的空间序结构、时间序结构和功能序结构的规律性。

例如，对于随机服务系统，顾客、队列和服务台等系统要素及其相互关系形成了系统的空间序结构，顾客到达时间、排队等待时间和服务时间等形成了系统的时间序结构。系统的功能，包括排队、对顾客的选择服务等，形成了系统的功能序结构。仿真就是首先用数学语言，然后用计算机程序语言描述该系统的序结构，由此形成的仿真模型与随机服务系统具有相似的序结构，仿真模型与实际系统就具备了序结构上的相似性。

(2) 信息原理

相似理论的信息原理认为,系统的序结构的形成和演化与系统的信息作用相关。当不同系统间的信息作用存在共同性时,系统间形成信息作用的相似性。信息作用的内容、形式和信息场强度及其分布规律越接近,系统间的特性越相似。基于系统相似性的仿真模型应反映系统的信息作用规律,包括信息作用的内容、形式和信息场强度及其分布规律。

例如,上述随机服务系统中,顾客到达时间间隔的概率分布可以看成是系统的信息之一,对应不同的分布规律产生不同的时间序结构。如果要对一个温度控制系统进行仿真,就必须把实际温度作为反馈信息与规定温度进行比较产生偏差信息,并转换成某种形式的控制信息,从而反映实际系统的功能序结构。多媒体仿真一般应具有听觉和视觉功能,因此必须考虑音响信息和图像信息的产生、传递和相互作用关系,相应地应设计音响和图像发生系统,其仿真系统的空间序结构和功能序结构与一般仿真相比较而言就有所不同,且与实际系统具有不同性质和不同层次的相似性。

(3) 支配原理

相似理论的支配原理认为,受相同自然规律支配的系统间存在一定的相似性。系统相似程度的大小取决于支配系统的自然规律的接近程度。基于系统相似性的仿真模型应能够反映支配实际系统的自然规律。因此,应研究这些自然规律,并以某种形式体现在仿真模型中。例如,人口增长、种群繁殖及新产品和新技术的推广在一定条件下具有指数增长规律或阻滞增长规律,都可以用相同形式的微分方程模型表示,据此建立的仿真模型反映了不同系统具有相同的动态变化过程。

1.2.2 基于相似理论的系统仿真

系统仿真模型与实际系统应具有某种形式和一定程度的相似性,相似理论的基本原理为相似模型的建立提供了理论基础和思想方法。这些原理的应用应贯穿于整个仿真过程中,即在建模和仿真的各个阶段始终要进行相似的定性和定量分析,在此基础上形成一套可操作的工程化的方法和技术,使系统仿真更加精确、高效和可信。

1. 定义被仿真系统

被仿真系统的定义是对仿真目的和仿真对象的概念性描述,主要以文字、逻辑关系图和数据表格等形式表述,即建立系统模型。

(1) 明确仿真目的

一个特定的系统仿真问题通常有一个有限的目的,因此,所建立的模型仅仅是实际系统的一个有限的映像,其所具有的相似特征是有限的,只能是在某种程度和某种形式上的相似。明确仿真目的有助于把握仿真过程中可能要用到的相似形式和相似方法以及仿真可信性评价的标准。总的来说,仿真目的是规定建模和仿真活动的方向。具体来说,如果仿真的目的是要在计算机上再现实际系统的行为,就要根据行为相似的原理,用数学相似的方法进行仿真。如果仿真的目的是要在仿真系统上实现实际系统的某些功能,就要考虑用功能相似的原理,用数学相似和物理相似的方法建立数学模型和物理效应模型去模拟实际系统的功能。另一方面,根据问题的重要性和实现的可能性,确定一定的评价标准,有助于把握仿

真的复杂性和精确性要求，并作为衡量仿真是否达到目的的定性或定量指标。

(2) 仿真对象的描述

根据仿真目的，用系统的观点，确定仿真对象的组成要素和系统边界、系统环境等。应尽可能多地收集与仿真对象有关的概念、原理和模型，以此作为先验知识。尽可能多地收集对仿真对象进行实际观测所得到的试验数据，根据相似理论，这些先验知识和试验数据作为信息源在建模者大脑中形成相似元，为其后的相似分析和推理奠定基础，并且可能作为仿真可信性评价的标准之一。

数学相似模型本质上是用数学语言，包括数学符号、数学表达式和逻辑关系图，来反映仿真模型与实际系统间信息传递规律的相似性。对于连续系统采用微分方程或偏微分方程，对于离散事件系统采用概率分布函数及逻辑关系图。物理相似模型本质上应反映仿真模型与实际系统间几何和物理性质上的相似性，特别是要反映实际系统的物理效应。

值得一提的是，对于某一具体问题，所在的学科领域已有许多成熟的理论、方法和技术，仿真模型的建立离不开这些理论、方法和技术的指导和应用。可以在相似理论的指导下进行综合，并可能创造出新的相似性实现的原理、方法和技术。

2. 进行相似分析

相似分析就是根据仿真目的和仿真对象，确定为实现仿真目的系统模型应具有的相似形式和相似特性，据此确定相似要素，研究相似规律和相似性的实现原理、方法和技术。

(1) 确定相似形式

根据基于相似理论的系统仿真的基本原理，以仿真目的为导向的仿真模型与实际系统具有某种形式的相似性。仿真模型的相似形式可分为数学相似、物理相似或数学-物理相似、行为相似或功能相似、状态相似或结构相似，等等。一个仿真模型可能兼有几种相似形式，要考虑各种相似形式的可实现性。

(2) 确定相似要素

在确定仿真对象的组成要素及仿真系统应具有的相似形式的基础上，根据同序结构原理，分析并确定仿真系统的相似要素。

(3) 确定相似特性

各个相似要素具有某些属性和特征，这些属性和特征反映了相似要素所具有的特性，根据相似理论，相似是指系统要素间的特性相似。因此，仿真系统应具有与实际系统的要素特性的相似性。根据仿真目的、相似形式以及相似要素属性和特征的重要程度，选择适当的属性，并确定其特征，从而构成仿真系统应具有的相似特性。

(4) 研究相似规律和相似性实现的原理、方法和技术

根据相似理论的信息原理和支配原理，具有相同信息作用规律和自然支配规律的系统间呈现出相似性。因此，相似规律包括信息作用规律和自然支配规律。应根据仿真目的和仿真对象的先验知识，找出反映系统外部行为特征的内在相似规律，在此基础上，根据仿真系统的相似形式以及具体问题领域已有的科学原理、方法和技术，确定实现相似性的原理、方法和技术。

3. 建立相似模型

在相似分析的基础上，建立某种形式的相似模型，即仿真建模。相似模型的形式根据仿真类型分为数学相似、物理相似和数学-物理相似。

数学仿真模型包括数学模型及对其进行有效求解的计算机。因此，要根据数学模型的特点和精度要求，选用适当的计算机系统及求解算法，用计算机程序语言表达数学模型。实际上，系统的数学模型是实际系统的一次近似，仿真模型则是实际系统的二次近似，本质上也是实际系统的相似模型。物理仿真模型不但应反映与实际系统间几何和物理性质上的相似性，更要反映实际系统的物理效应。数学-物理仿真模型兼有上述两者的要求，还要考虑两者之间的关联，包括实时通信、交互作用等。

4. 仿真模型运行及相似性分析

数字仿真本质上是用模型在计算机上进行试验，即在计算机上运行仿真模型。模型运行所得的结果即模型行为，包括点行为、轨迹行为和结构行为。其中一部分要与实际系统的观测结果进行比较，一部分可能是无法或尚未从实际系统中获得的结果，这也正是要进行仿真的原因之一。对于前者，模型行为与实际系统的观测结果虽然不可能完全一致，但应有一定的精度，才能反映出行为相似的特征，这是检验仿真模型和仿真过程是否具有实际系统的相似特征的最终途径。根据相似理论的基本原理，系统的序结构和信息作用具有可变性，应根据变动的序结构和信息作用来研究相似性的变化。因此，应改变输入信息和模型结构及其参数，对仿真模型进行多次运行。多次运行的结果除了用于相似性变化的分析外，还可达到结构和参数优化的目的。对于那些包含随机因素的模型，仅仅一次运行所得的行为结果是不充分的，需要多次运行。

1.2.3 系统仿真的类型

可以从不同的角度对系统仿真加以分类。比较典型的分类方法是：根据模型的种类分类；根据仿真所采用的计算机类型分类；根据仿真时钟与实时时钟的比例关系分类；根据系统模型的特性分类。

1. 根据模型的种类分类

根据模型的种类不同，系统仿真可分为三种：物理仿真、数学仿真和半实物仿真。

第一类称为物理仿真，即按照真实系统的物理性质构造系统的物理模型，并在物理模型上进行试验的过程。物理仿真的优点是直观、形象。在计算机问世以前，基本上是物理仿真，也称为“模拟”。物理仿真的缺点是模型改变困难，试验限制多，投资较大。

第二类称为数学仿真，即对实际系统进行抽象，并将其特性用数学关系加以描述而得到系统的数学模型，对数学模型进行试验的过程。计算机技术的发展为数学仿真创造了环境，使得数学仿真变得方便、灵活、经济，因而数学仿真亦称为计算机仿真。数学仿真的缺点是有时受限于系统建模技术，即系统的数学模型不易建立。

第三类称为半实物仿真，即将数学模型与物理模型甚至实物联合起来进行试验。对系

统中比较简单的部分或对其规律比较清楚的部分建立数学模型，并在计算机上加以实现；而对比较复杂的部分或对规律尚不十分清楚的系统，其数学模型的建立比较困难，则采用物理模型或实物。仿真时将两者连接起来完成整个系统的试验。

2. 根据仿真计算机类型分类

仿真技术是伴随着计算机技术的发展而发展的。在计算机问世之前，基于物理模型的试验一般称为“模拟”，它一般附属于其他相关学科。自从计算机特别是数字计算机出现以后，其高速计算能力和巨大的存储能力使得复杂的数值计算成为可能，数字仿真技术得到蓬勃发展。

按所使用的仿真计算机类型也可将仿真分为三类：模拟计算机仿真、数字计算机仿真和数字模拟混合仿真。

模拟计算机本质上是一种通用的电气装置，这是 20 世纪 50～60 年代普遍采用的仿真设备。将系统数学模型在模拟机上加以实现并进行试验称为模拟机仿真。本质上，模拟机仿真是一种并行仿真，即在仿真时，代表模型的各部件是并发执行的。

数字计算机仿真是将系统数学模型用计算机程序加以实现，通过运行程序来得到数学模型的解，从而达到系统仿真的目的。早期的数字计算机仿真则是一种串行仿真，因为计算机只有一个中央处理器(CPU)，计算机指令只能逐条执行。为了发挥模拟计算机并行计算和数字计算机强大的存储记忆及控制功能，以实现大型复杂系统的高速仿真，20 世纪 60～70 年代，在数字计算机技术还处于较低水平时，产生了数字模拟混合仿真，即将系统模型分为两部分，其中一部分放在模拟计算机上运行，另一部分放在数字计算机上运行，两个计算机之间利用模数和数模转换装置交换信息。

随着数字计算机技术的发展，其计算速度和并行处理能力的提高，模拟计算机仿真和数字模拟混合仿真已逐步被全数字仿真取代，因此，今天的计算机仿真一般指的就是数字计算机仿真。

3. 根据仿真时钟与实际时钟的比例关系分类

实际动态系统的时间称为实际时钟，而系统仿真时模型所采用的时钟称为仿真时钟。根据仿真时钟与实际时钟的比例关系，系统仿真可分类为：

(1) 实时仿真，即仿真时钟与实际时钟完全一致，也就是模型仿真的速度与实际系统运行的速度相同。当被仿真的系统中存在物理模型或实物时，必须进行实时仿真，例如各种训练仿真器就是这样，也称为硬件在回路中(hardware in the loop)仿真。

(2) 亚实时仿真，即仿真时钟慢于实际时钟，也就是模型仿真的速度慢于实际系统运行的速度。当对仿真速度要求不苛刻的情况下均是亚实时仿真，例如大多数系统离线研究与分析，有时也称为离线仿真。

(3) 超实时仿真，即仿真时钟快于实际时钟，也就是模型仿真的速度快于实际系统运行的速度。例如大气环流的仿真、交通系统的仿真等。

4. 根据系统模型的特性分类

仿真基于模型，模型的特性直接影响着仿真的实现。从仿真实现的角度来看，系统模型

特性可分为两大类，一类称为连续系统，另一类称为离散事件系统。由于这两类系统固有运动规律的不同，因而描述其运动规律的模型形式就有很大的差别，相应地，系统仿真技术也分为两大类：连续系统仿真和离散事件系统仿真。

(1) 连续系统仿真

连续系统是指系统状态随时间连续变化的系统。连续系统的模型按其数学描述可分为：①集中参数系统模型，一般用常微分方程(组)描述，如各种电路系统、机械动力学系统、生态系统等；②分布参数系统模型，一般用偏微分方程(组)描述，如各种物理和工程领域内的“场”问题。

需要说明的是，离散时间变化模型中的差分模型(参见表 1.1)可归为连续系统仿真范畴。原因在于，当用数字仿真技术对连续系统仿真时，其原有的连续形式的模型必须进行离散化处理，并最终也变成差分模型。

(2) 离散事件系统仿真

离散事件系统是指系统状态在某些随机时间点上发生离散变化的系统。它与连续系统的主要区别在于：状态变化发生在随机时间点上。这种引起状态变化的行为称为“事件”，因而这类系统是由事件驱动的；而且，“事件”往往发生在随机时间点上，亦称为随机事件，因而离散事件系统一般都具有随机特性；系统的状态变量往往是离散变化的。例如，电话交换台系统，顾客呼号状态可以用“到达”或“无到达”描述，交换台状态则要么处于“忙”状态，要么处于“闲”状态；系统的动态特性很难用人们所熟悉的数学方程形式(如微分方程或差分方程等)加以描述，而一般只能借助于活动图或流程图，这样，无法得到系统动态过程的解析表达。对这类系统的研究与分析的主要目标是系统行为的统计性能而不是行为的点轨迹。

模型可以被进一步划分为静态的或动态的、确定性的或随机的、定量的或定性的。静态模型通常被称为蒙特卡洛仿真，代表了在一个特殊时间点上的系统。动态模型代表了随时间变化的系统。不包含随机变量的模型被划分为确定性的，确定性模型具有一组已知的输入，并产生一组唯一的输出。随机模型具有一个或多个随机变量作为输入，随机输入导致随机输出，由于输出是随机的，所以只能被作为模型真实特性的估计。例如，银行的仿真通常包括随机到达时间间隔及随机服务时间。因此，在随机仿真中，输出的测量结果(平均等待人数、单个客户平均等待时间)必须作为系统真实特性的统计估计。

上述模型均可以称为定量的，即模型可以用具有“量”属性的描述变量定义，这些变量的值在某一时刻是确定的或在统计意义下是确定的。定性仿真(qualitative simulation)是以非数字手段处理信息输入、建模、行为分析和结果输出等仿真环节，通过定性模型推导系统的定性行为描述。

另外，离散仿真模型不总是用于对离散系统进行建模，连续仿真模型也不总是用于对连续系统进行建模。例如，尽管我们知道液体的流动是连续的，但一些软件销售商将储油箱和管道进行离散建模。对于离散或是连续(或既是离散也是连续)仿真模型的选择，是所研究的系统及对象的特性的函数。例如，如果认为每条消息的特性和运动都是重要的，通信信道就可以以离散的方式建模。反之，如果信道上的消息流集合是重要的，则通过连续仿真对系统建模可能更恰当。另外，仿真模型也可以是混合的，即既有离散的也有连续的。

1.2.4 系统仿真的一般步骤

系统仿真的一般步骤可用图 1.3 来描述。

正如前面所述，仿真是基于模型的活动，首先要针对实际系统建立其模型与形式化，一方面根据研究和分析的目的，确定模型的边界，因为任何一个模型都只能反映实际系统的某一部分或某一方面，也就是说，一个模型只是实际系统的有限映射。另一方面，为了使模型具有可信性，必须具备对系统的先验知识及必要的试验数据。特别是，还必须对模型进行形式化处理，以得到计算机仿真所要求的数学描述。模型可信性检验是建模阶段的最后一步，也是必不可少的一步。只有可信的模型才能作为仿真的基础。

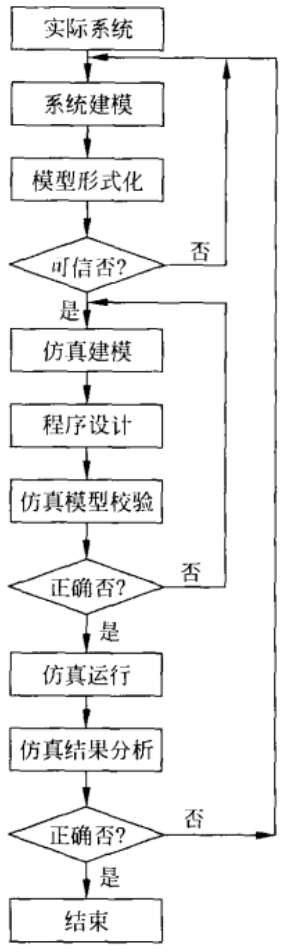


图 1.3 仿真的一般步骤

仿真建模是仿真过程的第二步。其主要任务是：根据系统的特点和仿真的要求选择合适的算法，当采用该算法建立仿真模型时，其计算的稳定性、计算精度、计算速度应能满足仿真的需要。

第三步是程序设计，即将仿真模型用计算机能执行的程序来描述。程序中还要包括仿真实验的要求，如仿真运行参数、控制参数、输出要求等。早期的仿真往往采用高级语言编程，随着仿真技术的发展，一大批适用不同需要的仿真语言被研制出来，大大减轻了程序设计的工作量。

仿真模型校验一般是不可缺少的。一方面是程序调试，更重要的是要检验所选仿真算法的合理性。这是仿真过程的第四步。

有了正确的仿真模型，就可以对模型进行试验，这是实实在在的仿真活动。它根据仿真的目的对模型进行多方面的试验，相应地得到模型的输出，这是第五步。

仿真过程的第六步是要对仿真输出进行分析。以往，输出分析的方法学未能引起人们的足够重视，实际上，输出分析在仿真活动中占有十分重要的地位，特别是对离散事件系统来说，其输出分析甚至决定着仿真的有效性。输出分析既是对模型数据的处理(以便对系统性能做出评价)，同时也是对模型的可信性进行检验。

上面我们仅仅对仿真过程的主要步骤进行了简要说明。在实际仿真时，上述每一个步骤往往需要多次反复和迭代。

1.3 仿真科学与技术的应用

仿真科学与技术已经有近 100 年的发展历史，它不仅用于航天、航空、各种武器系统的研制部门，而且已经广泛应用于电力、交通运输、通信、化工、核能各个领域。特别是，近 20 年来，随着系统工程与科学的迅速发展，仿真技术已从传统的工程领域扩展到非工程领域，

因而在社会经济系统、环境生态系统、能源系统、生物医学系统、教育训练系统也得到了广泛的应用。仿真科学与技术正是从其广泛的应用中获得了日益强大的生命力，而仿真技术的发展反过来使其得到愈来愈广泛的应用。

在系统的规划、设计、运行、分析及改造的各个阶段，仿真科学与技术都可以发挥重要作用。随着人类所研究的对象规模日益庞大，结构日益复杂，仅仅依靠人的经验及传统技术难以满足愈来愈高的要求。基于现代计算机及其网络的仿真科学与技术，不但能提高效率，缩短研究开发周期，减少训练时间，不受环境及气候限制，而且对保证安全、节约开支、提高质量尤其具有突出的功效。

1.3.1 仿真在系统设计中的应用

系统设计是一项复杂的任务，计算机辅助设计及仿真技术为系统设计提供了强有力的工具。

一个较为复杂的系统，其设计过程一般要经历可行性论证、初步设计、详细设计、实施等若干阶段。在每个阶段，仿真技术均可提供强有力的技术支持。在可行性论证阶段，可以根据系统设计的目标及边界条件，对各种方案进行定量比较，发现不同方案的优缺点，做到“心中有数”，真正了解方案为什么“可行”或“不可行”，为系统设计打下坚实的基础。在系统设计阶段，设计人员可以利用仿真技术建立或完善系统模型，进行模型试验、模型简化并进行优化设计，因而国内外开发的许多计算机辅助设计软件大都包含仿真子包。

系统设计中经常涉及新的设备、部件或控制装置，此时，可以利用仿真技术进行分系统试验，即一部分采用实际部件，另一部分采用模型。这样，既可避免由于新子系统投入可能造成的对原系统的破坏或影响，又可大大缩短开工周期，提高系统投入的一次成功率。例如，我国陡河电站25万千瓦发电机组的安装，由于事先在电厂仿真系统上已进行了细致的分系统试验，对全部自动装置的参数都做了整定，在实际机组安装完毕的同时，自动装置也全部调试完成，很快地投入了运行。而按一般程序，机组安装完后，现场调试自动装置的参数，周期大约需要一年。

随着科学技术的发展，系统的规模日趋庞大，系统的结构日益复杂，系统设计方案的正确性与合理性检验愈来愈困难，结果给系统的实施带来了巨大的风险。例如，从20世纪80年代出现的计算机集成制造系统(CIMS)，它要实现企业在计算机网络及数据库的支持下，从市场预测、产品设计、生产计划、库存控制、成本控制、制造加工到销售服务全部过程的集成优化。显然，这是一件十分困难的任务，仅仅依靠人的经验难以达到预期的效果。因此，仿真技术成为设计这类系统的不可缺少的工具。例如，美国在研究设计其AMRF(advanced manufacturing research facility)过程中，广泛采用了仿真技术，并开发了专用仿真系统HCSE(hierarchical control system emulator)以用于AMRF的递阶控制系统的设计、开发、调试及测试。

1.3.2 仿真在系统分析中的应用

要对系统进行分析，就必须对系统进行试验，通过试验来了解系统的结构及其内部发生的活动，从而达到对系统的正确评价。有两种试验方案：一种是直接在真实系统上进行，如

飞机试飞、轮船试航、汽车试车等。通过试验，发现设计或制造中的技术或工艺问题，以便在正式投产前或投入运行前加以改正。另一种试验是按实际系统构造模型，对模型进行仿真试验，即仿真分析。尽管在真实系统上进行试验在许多情况下仍然是必不可少的，但是由于下述原因，仿真试验与分析越来越普遍地被采用。(1)在真实系统上试验会破坏系统的正常运行；(2)由于实际系统中各种客观条件的限制，难以按预期的要求改变参数，或者得不到所需要的试验条件；(3)在实际系统上进行试验时，很难保证每一次的操作条件都相同，因而难以对试验结果的优劣做出正确的判断和评价；(4)无法复原；(5)试验时间太长、费用太大或者有危险等。

1.3.3 仿真在教育与训练中的应用

一般说来，凡是需要有一个或一组熟练人员进行操作、控制、管理与决策的实际系统，都需要对这些人员进行训练、教育与培养。早期的培训大都是在实际系统或设备上进行的。随着系统规模的加大、复杂程度的提高，特别是造价日益昂贵，训练时因操作不当引起破坏而带来的代价大大增加，因此，提高系统运行的安全性事关重大。以发电厂为例，美国能源管理局的报告认为，电厂的可靠性可以通过改进设计和加强维护来改善，但只能占提高可靠性的20%～30%，其余要依靠提高运行人员的素质来提高，可见，人员训练对这类系统的重要性。为了解决这些问题，需要有这样的系统，它能模拟实际系统的工作状况和运行环境，又可避免采用实际系统时可能带来的危险性及高昂的代价，这就是训练仿真系统。

训练仿真系统是利用计算机并通过运动设备、操纵设备、显示设备、仪器仪表等复现所模拟的对象行为，并产生与之适应的环境，从而成为训练操纵、控制或管理这类对象的人员的系统。

根据模拟对象、训练目的，可将训练仿真系统分为三大类型：

(1) 载体操纵型　这是与运载工具有关的仿真系统，包括航空、航天、航海、地面运载工具，以训练驾驶员的操纵技术为主要目的。

(2) 过程控制型　用于训练各种工厂(如电厂、化工厂、核电站、电力网等)的运行操作人员。

(3) 博弈决策型　用于企业管理人员(厂长、经理)、交通管制人员(火车调度、航空管制、港口管制、城市交通指挥等)和军事指挥人员(空战、海战、电子战等)的训练。

我国在研制各类训练仿真系统方面已经取得了不少成果，如用于飞机起落训练的飞行仿真器，用于舰船进出港训练的船舶操纵训练仿真器，用于电厂运行人员训练的电厂训练仿真器，以及用于海战训练的海军战术训练仿真器。

近年来，分布交互式训练仿真系统得到广泛的注意。这类系统将分布在不同地点、已存在的各种不同类型的训练仿真系统，通过计算机网络进行集成，从而实现更大规模综合训练。典型的是美国SIMNET，它将分布在美国和德国11个城市的260个地面装甲车辆仿真器和飞行模拟器集成起来，形成一个广域战场，进行多兵种合成训练。

1.3.4 仿真在产品开发及制造过程中的应用

进入20世纪90年代以来，“虚拟产品开发”(virtual product development)技术引起了

人们的广泛关注，近几年来，人们又进一步提出了虚拟制造（virtual manufacturing）的概念。

虚拟制造是实际制造过程在计算机上的本质实现，即采用计算机仿真与虚拟现实技术，在计算机上群组协同工作，在计算机上建立产品的三维全数字化模型，“在计算机上制造”产生许多“软”样机，从而，在设计阶段，就可以对所设计的零件甚至整机进行可制造性分析，这包括加工过程的工艺分析、铸造过程的热力学分析、运动部件的运动学分析以及整机的动力学分析等，甚至包括加工时间、加工费用、加工精度分析等。设计人员或用户甚至可“进入”虚拟的制造环境检验其设计、加工、装配和操作，而不依赖于传统的原型样机的反复修改。这样使得产品开发走出主要依赖于经验的狭小天地，发展到了全方位预报的新阶段。

图 1.4 简要表示了虚拟制造与实际制造的联系与区别。可以说，虚拟制造就是实际制造在计算机上的本质实现，是仿真技术以制造过程为对象的全方位的应用。

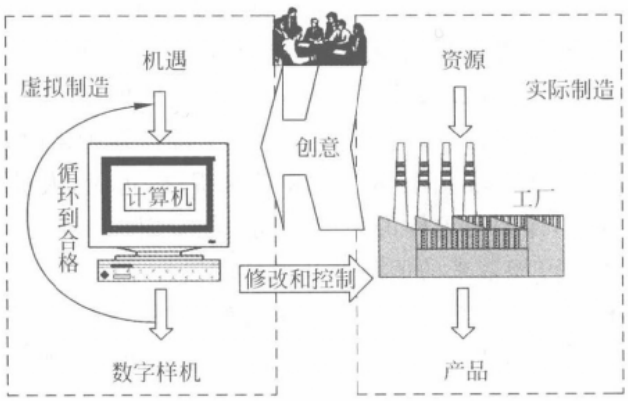


图 1.4 虚拟制造与实际制造

典型的虚拟制造例子是波音 777 的研制，其整机设计、部件测试、整机装配以及各种环境下的试飞均是在计算机上完成的，其开发周期从过去的 8 年缩短到 5 年。又如 Perot System Team 利用 Dench Robotics 开发的 QUEST 及 IGRIP 设计与实施一条生产线，在所有设备订货之前，对生产线的运动学、动力学、加工能力等各方面进行了分析与比较，使生产线的实施周期从传统的 24 个月缩短到 9.5 个月。Chrysler 公司与 IBM 合作开发的虚拟制造环境用于其新型车的研制，在样车生产之前，发现其定位系统的控制及其他许多设计缺陷，缩短了研制周期。

1.4 当前仿真科学与技术研究的热点

1.4.1 网络化仿真技术

网络化仿真技术泛指以现代网络技术为支撑实现系统建模、仿真运行试验、评估等活动的一类技术。现代网络化建模仿真技术始于分布交互仿真技术。20 世纪 80 年代初期，在

军事应用与 Internet 技术发展的推动下，出现了分布仿真系统，它受当时网络技术的限制，在可扩展性、性能及其应用方面存在着许多局限性。为满足对逼真度更高的分布仿真的迫切需求，20 世纪 80 年代中期，美国国防部提出了先进分布仿真(ADS)技术的概念，美国国防高级研究计划局(DARPA)和陆军率先共同制订了 SIMNET(SIMulator NETworking)计划。自此，开辟了 ADS 技术发展的新纪元，并经历了 DIS、ALSP 和 HLA 等几个典型的发展阶段。HLA 在军事上得到了许多成功的应用，与此同时，HLA 在工业上亦引起广泛重视，已成为 IEEE 1516 标准。

HLA 只是分布交互仿真技术发展的新起点，还存在某些局限性，它随着仿真需求、仿真技术和各种支撑技术的发展不断地得到发展。特别是，随着互联网、Web/Web Service、网格计算(grid computing)等网络技术的发展，其技术内涵和应用模式得到不断地扩展和丰富，并在下面几个方面成为今天网络化仿真研究应用关注的焦点。

(1) 基于 Web 技术的可扩展建模仿真框架 XMSF

基于 Web 的仿真可追溯到 WWW 的产生，但是该概念的正式提出是在 1996 年的冬季仿真会议上。Web-based Simulation 初期的研究主要是针对 WWW 新生技术对仿真领域的影响，以及基于 Web 的仿真的研究内容。随着 Internet/Web 技术的不断发展，2000 年以后，M&S 与 Web/XML，Internet/Networking 技术的进一步结合推动形成了 XMSF (extensible modeling and simulation framework)。XMSF 的核心是使用通用的技术、标准和开放的体系结构促进建模与仿真在更大范围(尤其包括与 C4ISR 系统)的互操作与重用。当前基于 Web 技术的网络化建模仿真技术研究发展的重点是 XMSF 相关标准、技术框架和应用模式等的建立与完善，HLA 中 SOM/FOM 与 BOM(based object model)的结合和 HLA/RTI 的开发应用与 Web/Web Service 技术的结合，实现在公共广域网条件下的联邦组织和运行。其重要意义在于推动分布建模仿真技术向着标准化、组件化，以及仿真嵌入实际系统的方向发展。

(2) 基于网格计算技术的仿真网格技术

网格技术是一项新兴并且正在发展的技术，其核心是解决网上各种资源(如计算资源，存储资源，软件资源，数据资源等)的动态共享与协同应用。网格与仿真的结合为各类仿真应用对仿真资源的获取、使用和管理提供了巨大的空间，同时，它以崭新的理念和方法为仿真领域中诸多挑战性的难题的解决提供了技术支撑(如仿真应用的协同开发，仿真运行的协调、安全和容错，模型和服务的发现机制，资源管理、监控和负载平衡等等)。

仿真网格以应用领域仿真的需求为背景，综合应用复杂系统模型技术、先进分布仿真技术、虚拟现实技术、网格技术、管理技术、系统工程技术及其应用领域有关的专业技术，实现网格/联邦中各类资源(包括仿真系统/项目参与单位有关的模型资源、计算资源、存储资源、网络资源、数据资源、信息资源、知识资源、软件资源，与应用相关的物理效应设备及仿真器等)安全地共享与重用、协同互操作、动态优化调度运行。

1.4.2 复杂系统/开放复杂巨系统的建模与仿真

复杂系统广泛存在于各个领域，如宏观经济系统、社会系统、军事系统、地理系统、生物系统、交通系统、制造系统、网络系统等。关于复杂系统目前尚无普遍认同的定义，一般说

来，复杂系统除了具有一般系统的特点外，还具有大规模、非线性、变结构、自治性、开放性、不确定性等特点，其功能、行为复杂并难以预测等。

(1) 复杂系统往往规模很大，而且有大量因素、组成部分相互作用。虽然拉普拉斯有一句名言：只要建立一个详尽的微分方程，并给出初始条件，就能预测世界上任何时候的任何现象。然而这个名言却难以验证。举例说，在一个 $1cm^3$ 的容器里，有大约 2.7×10^{19} 个气体分子。假设每个分子的运动都符合已知的力学定律，若要计算全体分子运动的总效果，即使能建立其数学模型，目前也难以解算规模如此巨大、变量如此众多的方程组。现代的流体力学就将其假设为连续介质加以处理。

(2) 复杂系统往往具有强非线性。真实系统绝大多数都是非线性的，线性系统只是一种近似。至今人们广泛采用的科学方法大多适用于线性系统，而非线性系统蕴含着的复杂性是线性系统不可比拟的。混沌学研究表明，相当多的非线性系统，初始条件的微小差异可导致结果的巨大差异。

(3) 复杂系统往往表现为模式的多样性、变化的跳跃性、相变的突然性。系统的结构往往在多种因素的作用下发生融合、转化、演化或衍生，具有不同类型的变结构性和开放性。

(4) 复杂系统还往往具有不确定性(包括随机性)。不确定性系统的行为特性显然较确定性系统要复杂得多。虽然概率统计方法对随机系统的研究提供了有力工具，但这种方法处理的是系统的宏观行为，却无法处理系统内部个体本质机制。模糊数学也是以“人为简化”的方法来处理不确定性，目前尚处于不成熟阶段。

早在 20 世纪 80 年代，复杂系统就已经引起人们的关注，其中以诺贝尔物理学奖获得者盖尔曼(Murray Gell-Mann)和安德逊(Philip Anderson)、经济学奖获得者阿诺(Kenneth Arrow)等人为代表，一批从事物理、经济、理论生物、计算机学科的研究人员于 1984 年成立圣塔菲研究所(Santa Fe Institute,SFI)，专门从事复杂性科学研究，探索通过学科间交叉与融合来解决复杂性问题。

仿真被人们认为是信息时代与理论研究、实验研究并驾齐驱的第三种认识世界的方法，因此复杂系统建模与仿真技术将成为研究各类复杂系统的最重要手段。当前研究的热点包括：

(1) 定性仿真(qualitative simulation)

定性仿真有其独到之处，这种仿真能处理多种形式的信息，有推理能力和学习能力，能初步模仿人类思维方式，人机界面更符合人的思维习惯，所得结果更容易理解。目前，定性仿真基本可分为三类，即模糊数学方法、归纳推理方法和朴素物理方法。模糊数学方法可以解决模型信息与测量数据的不确定性，所以在定性理论中一般用来作为一种描述手段。归纳推理方法起源于通用系统理论，主要利用其中的通用系统问题求解(general system problem solving)技术，输入尽可能多的行为，通过归纳学习的方式，构造系统的定性模型，进行仿真研究。朴素物理方法在理论和应用上发展得最为成熟，它兴起于一些人工智能专家对朴素物理系统的定性推理研究。

关于定性仿真研究的热点包括：①定量与定性结合的仿真。由于定性模型中包含系统的不完全知识，定性仿真会产生一些虚假和二义的多余行为，当实际系统很复杂时，定性仿真产生相当数量的多余行为，如何有效地减少定性仿真产生的行为数，成为当今定性推理研究的主题。很多研究者纷纷采用定量与定性结合的仿真方法。在定性仿真中加入相当的定

量知识，将定量与定性有机地结合起来，将大大减少系统的预测行为数，增强定性仿真的生命力。②模型分解方法研究。定性仿真走向应用时，往往涉及规模较大的系统，即使省略某些细节，模型仍是非常复杂的。所以，定性理论中，必须有处理这种复杂性的手段。模型分解方法将系统模型分为若干部分，称为组件(component)，系统的联系紧密的变量将集中在一个组件中，并为组件建立状态，系统的描述将以这种状态为单位，若需要不同部分的变量的事件对应性，可以通过不同部分之间的连接来产生。并且，仿真算法上也作了相应的变动，以局部的组件描述为基础的仿真取代了以全局状态为基础的定性仿真算法。大大提高了模型建立工作的效率和准确性，并降低了仿真的时间和空间运行代价。③并行定性仿真。定性仿真在减少冗余或虚假行为的研究上取得了很大进展，但同时也带来了一些始料未及的副作用：定性与定量知识的结合，使知识的表示和推理机制复杂化，数据量明显增加；由于信息不完备，系统的搜索空间增大，使得定性仿真在一定的情况下比定量仿真的速度更慢；再者随着定性仿真逐渐走向应用，参数数量的增长使问题的规模呈指数增长，仿真的速度也明显下降。而并行定性仿真能较大幅度地提高定性仿真的效率，因此成为研究的热点。

(2) 基于 Agent 的仿真

20 世纪 90 年代提出 Agent 以来，它已经成为计算机领域与人工智能领域研究的热点。尽管到目前为止尚未有关于 Agent 的统一认同的定义，但已广泛应用于许多领域。人们普遍认为，Agent 具有自治性、社会性、反应性。自治性是指 Agent 的行为是主动的、自发的(至少有一种行为是这样的)，Agent 有它自己的目标或意图，能对自己的行为做出计划。社会性是指单个 Agent 的行为必须遵循和符合 Agent 社会的社会规则，并能通过某种交互语言以合适的方式进行 Agent 之间的交互与合作。反应性是指 Agent 能感知其所处的环境(可能是物理世界、人机交互界面或其它 Agent)，并能及时做出反应。在某些情况下，还可能赋予 Agent 其它特征，如理性(不会故意阻止其目标的实现等)、诚实性(不会故意传播虚假信息等)、友好性(总是努力完成其它 Agent 的请求等)。总之，Agent 具有某种程度的智能，这是它区别于计算机领域人们熟悉的概念“对象”的主要特点。

在仿真领域，面向 Agent 的仿真是一段时间以来的研究热点。基于 Agent 的社会性和自治性能力，通过 Agents 之间的异步信息传输，完成基于 Agent 对象的社会行为和社会反应的仿真，已经得到广泛有效的应用，特别是基于多 Agent 系统(multi Agent system，MAS)仿真被广泛应用于复杂系统的仿真中。因为它具有以下特点：

① 模块化　Agent 技术完全拥有面向对象技术所具有的模块化的能力，而且最适合于被分解成自然模块的应用问题。一个 Agent 拥有自己的状态变量集，能够与环境变量区分开来，它与环境的接口可以清楚地被识别，从而能很好地用于描述建模对象，还可以通过最小限度的改变被重用。

② 分散化　Agent 不需要外部激励就可以监视其环境，并可在它适合的时候采取行动，从而很方便地用于分布式仿真。

③ 可变性　同时具有模块化和分散化能力的 Agent 可很方便地用于结构变化频繁的系统仿真。因为模块化允许方便地修改模型的局部，分散化使得一个模块的改变对其它模块的影响很小。

④ 易于处理不良结构问题　不良结构是指在模型设计时，并非所有结构信息已知。

Agent 可以支持这样的系统模型设计，因为在 MAS 中，每个 Agent 只需自我信息和环境交互信息，至于环境内部如何，它无须了解。基于 Agent 的模型设计仅需要识别环境对它的影响和它对环境的影响，而不需细化相互连接的单个实体及它们之间的接口。

(3) 基于系统动力学的仿真

系统动力学理论强调系统、整体的观点和联系、发展、运动的观点。从系统方法论来说，系统动力学的方法是结构方法、功能方法和历史方法的统一，适合于宏观分析社会、经济、生态和生物等一类复杂大系统。

系统动力学研究处理复杂系统问题的方法是定性与定量结合、系统分析与综合推理的方法。系统动力学模型是一种结构-功能模型。对于以良结构为主的系统动力学模型，其解决问题的过程实质上是寻优过程，可用来寻求系统的较优结构与参数，以获取较优的系统功能。然而，一般而言，系统动力学的建模过程就是一个学习、调研的过程，其模型的主要功用在于向人们提供一个进行学习与仿真分析的对象工具。

按照系统动力学的理论与方法建立的模型，借助计算机仿真可以用于定性与定量地研究系统问题。系统动力学仿真是一种结构-功能的仿真；它最适用于研究复杂系统的结构、功能与行为之间的动态关系，一个突出的优点在于它能处理高阶次、非线性、多重反馈复杂时变系统的问题。目前系统动力学仿真已经在以下领域取得很好的进展：

① 系统动力学建模与层次分析法(AHP)、主成分分析法结合，用于对系统结构层次与横向之间关系的分析和主要变量的选择。

② 系统动力学与数量经济学、计量经济学相结合，用于经济系统中描述经济变量的定量关系。

③ 灰色系理论与方法为系统动力学模型中某些辅助方程的参数作估计，运用优化理论对模型进行结构、参数的次优化。

④ 非平衡自组织理论与系统动力学相结合，前者用于一组系统动力学模型进行同态简化，获得集中模型体模型；或者先建立系统动力学模型，推出系统演化方程，运用协同学找出序参数以解释系统的自组织现象。

系统动力学主要用于研究复杂系统的变化趋势，而不注重于数据精确性的领域。

1.4.3 综合自然环境的建模与仿真

许多仿真系统需要考虑自然环境条件的影响，以获得更符合实际情况的仿真结果；还有一些仿真系统直接以环境内部变化、环境效应和环境影响为研究目的。综合自然环境的建模与仿真包括对地理(地形、地貌和地质)、海洋、空间、大气、电磁等环境信息的建模与仿真。环境建模的目的是获取实际环境的多维数据，并根据应用需要建立相应的虚拟环境模型。在环境模型基础上进行的仿真应用则体现了自然环境对实体运行和决策行为产生的影响。综合自然环境的建模采用静态或动态的多维数据场拟合方法，并对虚拟自然环境与实体的交互进行检测。

自然环境建模与仿真技术现已广泛应用于战场模拟、军事训练、空间技术研究、工业检测与评估、多学科研究等领域。借助高性能计算机和传感器、计算机辅助设计技术(CAD)以及虚拟现实(virtual reality)技术，自然环境建模呈现视觉、听觉、触觉等多维的表现形式，

自然环境建模仿真实现了将环境中仿真个体与真实世界或虚拟世界进行有机的结合并能较真实地体现两者的交互行为。随着电子战、电力系统、数字通信和电子应用领域的逐渐发展，电磁环境建模与仿真也得到了相应的重视，包括电磁兼容性(EMC)仿真、电磁暂态实时仿真、无线频谱以及高频与微波环境仿真等。当前，自然环境建模与仿真技术向着多学科融合、实时动态化和分布式协同化方向发展，同时出现了相关的设计标准(综合环境数据表示与交换标准，SEDRIS)、规范和实现技术(地理信息系统 GIS，基于图像绘制 IBR，体绘制 volume rendering 等)。其中 SEDRIS(synthetic environment data representation and interchange specification)是旨在提供一个包括陆地、海洋、大气和外层空间的完整的物理环境模型以及相关的标准化数据的存取和交换方法。它提出了综合环境的数据模型，为综合环境数据库之间进行确定而损失最小的数据交换提供了统一的和有效的机制。

综合自然环境建模与仿真进一步研究的重点包括自然环境特征量的提取技术、多维数据场的动态建模与数据表示技术、虚拟自然环境的用户界面和人机交互技术、综合自然环境的多分辨率建模与模型压缩简化技术、自然环境内部建模和虚拟自然环境接口建模技术、分布及协同式虚拟自然环境技术、虚拟自然环境中特征识别与多维导航技术、综合自然环境中的非视觉可视化技术、虚拟自然环境运行的实时性技术、基于图像的虚拟环境绘制技术和体绘制技术以及综合自然环境建模与仿真的规范等。

1.4.4 基于普适计算技术的普适仿真技术

普适计算技术是将计算技术与通信技术、数字媒体技术相融合的技术，它提供一种全新的计算模式，其目标是使由计算和通信构成的信息空间与人们生活的物理空间相融合的智能化空间(smart space)，在这个智能化空间中人们可以随时随地透明地获得计算和信息服务。普适计算的思想最早是 1991 年 Mark Weiser 在《Scientific American》的“The Computer for the 21st Century”中提出的，其研究在学术上受到充分重视的重要标志是分别从 1999 年和 2000 年开始的 Ubicomp 国际会议和 Pervasive Computing 国际会议、2002 年 IEEE Pervasive Computing 期刊的创刊。当前普适计算在技术研究、编程环境开发和应用上都取得较大研究进展。

在仿真系统中引进普适计算技术，是将计算机硬软件、通信硬软件、各类传感器、设备、模拟器紧密集成，实现将信息空间与物理空间结合的一种新仿真模式，其重要意义是实现仿真进入真实系统，无缝地嵌入到我们的日常生活事物中。当前，普适仿真相关的重要研究内容涉及融合基于 Web 的分布仿真技术、网格计算技术、普适计算技术的先进普适仿真体系结构，信息空间和物理空间的协调管理和集成技术，基于普适计算的普适仿真自组织性、自适应性和高度容错性，以及普适仿真应用技术等。

普适计算技术与网格计算技术、Web service 技术的结合，将极大地拓展以人为本，对环境敏感、随时随地获取计算能力的智能化空间的应用边界，是未来 IT 的重要支撑。人们认为，融合普适计算技术、网格计算技术与 Web service 技术的“普适化仿真技术”将推动现代建模与仿真技术的研究、开发与应用进入一个崭新的时代。

1.4.5 基于高性能计算机的仿真

随着仿真应用的发展，特别是在广泛应用复杂数学模型的领域如天文、气象、生物、医学、能源等，单处理器往往不能满足需求，因此高性能计算机在仿真中的应用成为仿真科学与技术的另一热点。

有别于“通用计算机”，专门针对高性能科学计算/仿真进行优化设计的计算机开始被称为“高性能计算机”，或简称 HPC。20 世纪 70 年代出现的向量计算机可以看作是第一代的高性能计算机，通过在计算机中加入向量流水部件，可以大大提高科学计算中向量运算的速度。20 世纪 90 年代初期，大规模并行处理(MPP)系统已经开始成为高性能计算机发展的主流，MPP 主要由多个微处理器通过高速互联网络构成，每个处理器之间通过消息传递的方式进行通信和协调。较 MPP 早几年问世的对称多处理机 SMP 系统，是由数目相对较少的微处理器共享物理内存和 I/O 总线形成的计算机系统，它是单机系统的升级与增强，被广泛应用于商业计算领域。20 世纪 90 年代中后期的一种趋势是将 SMP 的优点和 MPP 的扩展能力结合起来，这一趋势发展成后来的 CC-NUMA 结构，即分布式共享内存。在发展 CC-NUMA 的同时，机群系统(cluster)也迅速发展起来。类似 MPP 结构，机群系统由多个微处理器构成的计算机节点通过高速网络互连而成。节点一般是可以单独运行的商品化计算机。由于规模经济成本低的原因，机群系统具有比 MPP 更高的性能/价格比优势。

高性能计算机的实现技术包括计算数学(计算模型与算法)、计算机体系结构与部件构成技术三部分。目前，高性能计算技术主要有两个发展方向：在深度上致力于研制具有快速运算能力的高性能计算机；在广度上注重于开发具有广泛应用前景的高性能服务器。

高性能计算机的发展趋势主要表现在网络化、体系结构主流化、开放和标准化、应用的多样化等方面。网络化的趋势将是高性能计算机最重要的趋势，高性能计算机的主要用途是网络计算环境中的主机，以后越来越多的应用是在网络环境下的应用，会出现数以十亿计的客户端设备，所有重要的数据及应用都会放在高性能服务器上，一个重要的趋势是超级服务器正取代超级计算机而成为高性能计算的主流体系结构技术。Client/Server 模式会进入到第二代，即服务器聚集的模式。

1.5 小结

本节是关于系统仿真的概述，首先对系统仿真中的三个基本概念——系统、模型、仿真——进行了定义，说明了概念的内涵及其相互关系。相似理论是仿真的理论基础之一，为此对其进行了原理性介绍，说明了系统仿真应遵循的基本原则，给出了系统仿真的一般步骤。本节还对仿真科学与技术在系统设计、分析、产品开发与制造、教育与训练等方面的应用进行了讨论，并对当前仿真科学与技术的研究热点进行了综述。

经过近一个世纪的发展历程，“仿真科学与技术”已形成独立的知识体系，包括：由仿真

建模理论、仿真系统理论和仿真应用理论构成的理论体系；由自然科学的公共基础知识、各应用领域内的基础专业知识和仿真科学与技术的基础专业知识综合而构成的知识基础；由基于相似理论的仿真建模，基于整体论的网络化、智能化、协同化、普适化的仿真系统构建和全系统、全寿命周期、全方位的仿真应用思想综合而成的方法论。“仿真科学与技术”已成为人类认识与改造客观世界的重要方法，在关系国家实力和安全的关键领域，如航空、航天、信息、生物、材料、能源、先进制造、农业、教育、军事、交通、医学等领域，发挥着不可或缺的作用。

习题

1. 使用搜索引擎，以关键词“仿真”、“simulation”搜索，整理相关资料，列出有关它们的定义，比较这些定义的差别。

2. 使用搜索引擎，以关键词“建模”、“modeling”搜索，整理相关资料，列出有关它们的定义，比较这些论述，给出你对“建模”、“modeling”与仿真的关系的描述。

3. 试述你对“仿真是基于模型的活动”的理解。

4. 相似理论包括的基本原理是什么？如何利用这些原理指导仿真活动的各个阶段？

5. 仿真过程中，模型校验（verification）与模型可信性确认（validation）为什么非常重要？

6. 使用搜索引擎，搜索“基于 Web 的仿真”（web based simulation）的网页，并进行综述。

参考文献

[1] Law A M, and Kelton W D. Simulation Modeling and Analysis. 3rd ed. New York: McGraw-Hill, 2000

[2] Sci-Tech Encyclopedia. Model theory. http://www.answers.com/topic/model-theory

[3] Sprier J A 等著. 王正中等译. 计算机辅助建模与仿真. 北京：科学出版社，1991

[4] Paul Fishwick Computer Simulation: The Art and Science of Digital World Construction, http://www.cis.ufl.edu/~fishwick/introsim/nodel.html, (Thu Oct 19 10: 30: 41 EDT 1995)

[5] 周美立. 相似学. 北京：中国科学技术出版社，1993

[6] 文传源. 系统仿真学科与仿真系统技术. 系统仿真学报，1992，4(3)：1～8

[7] 徐 迪. 基于相似理论的系统仿真方法. 系统工程理论与实践，2001，2：113～116

[8] 李文伟. 定性仿真方法发展与应用研究. 系统仿真技术，2008，4(2)：71～74，84

[9] Banks J, Carson J S 著. 肖田元，范文慧译. 离散事件系统仿真. 北京：机械工业出版社，2007

[10] 肖田元等. 虚拟制造. 北京：清华大学出版社，Springer 出版社，2004

[11] Li Bohu, Chai Xudong, Zhu Wenhai, Di Yanqiang, et al. Research and Implementation on Collaborative Simulation Grid Platform, SCSC, San Jose, USA, 2004

[12] Foster I. The Grid: A New Infrastructure for 21st Century Science. Physics Today, 2002, 55 (2): 42～47

[13] Donald Brutzman, Michael Zyda J, Mark Pullen and Katherine L. Morse: Extensible Modeling and Simulation

Framework (XMSF): Challenges for Web-Based Modeling and Simulation, Findings and Recommendations Report of the XMSF Technical Challenges Workshop and Strategic Opportunities Symposium, October 2002, http://www.movesinstitute.org/xmsf/XmsfWorkshopSymposiumReportOctober2002.pdf

[14] DS-Grid: Large Scale Distributed Simulation on the Grid e-Science Sister Project. http://www.mesc.bham.ac.uk, http://www.cs.bham.ac.uk/research/ds-grid

[15] The SEDRIS web site: www.sedris.org

[16] Dertouzos M L. The future of computing. Scientific American, 1999, 281(2): 52～55

[17] 樊建平,陈明宇.高性能计算机研究的现状与展望.信息技术快报,中国科学院计算技术研究所,2003,第5期

[18] 王其藩.系统动力学理论与方法的新进展.系统工程理论方法应用,1995,2: 6～12

第一篇　连续系统仿真篇

第 2 章
连续系统模型

按系统模型的特征分类，可以有连续系统仿真及离散事件系统仿真两大类。本篇讨论连续系统仿真问题。过程控制系统、调速系统、随动系统等称作连续系统，它们的共同之处是系统状态变化在时间上是连续的，可以用方程式（常微分方程、偏微分方程、差分方程）描述系统模型。

2.1 连续系统模型描述

本章讨论集中参数的连续系统，分布参数连续系统将在第 6 章中讨论。集中参数连续系统仿真中的数学模型描述形式有很多种，但基本上可分为三类：连续时间模型、离散时间模型及连续-离散混合模型。

2.1.1 连续时间模型

如果一个系统的输入量 $u(t)$，输出量 $y(t)$，系统的内部状态变量 $x(t)$ 都是时间的连续函数，那么我们可以用连续时间模型来描述它。系统的连续时间模型通常可以有以下几种表示方式：常微分方程、传递函数、权函数和状态空间描述。

1. 常微分方程

常微分方程可以表示为

$$a_0\frac{\mathrm{d}^n y}{\mathrm{d}t^n}+a_1\frac{\mathrm{d}^{n-1}y}{\mathrm{d}t^{n-1}}+\cdots+a_{n-1}\frac{\mathrm{d}y}{\mathrm{d}t}+a_n y=c_0\frac{\mathrm{d}^{n-1}u}{\mathrm{d}t^{n-1}}+c_1\frac{\mathrm{d}^{n-2}u}{\mathrm{d}t^{n-2}}+\cdots+c_{n-1}u \quad (2.1)$$

其中 n 为系统的阶次，$a_i(i=0,1,2,\cdots,n)$ 为系统的结构参数，$c_j(j=0,1,2,\cdots,n-1)$ 为输入函数的结构参数，它们均为实常数。

2. 传递函数

若系统的初始条件为零，那么对式(2.1)两边取拉氏变换后可得

$$a_0 s^n Y(s) + a_1 s^{n-1} Y(s) + \cdots + a_{n-1} sY(s) + a_n Y(s)$$
$$= c_0 s^{n-1} U(s) + c_1 s^{n-2} U(s) + \cdots + c_{n-1} U(s)$$

稍加整理，并记

$$G(s) = \frac{Y(s)}{U(s)} = \frac{\sum_{j=0}^{n-1} c_{n-j-1} s^j}{\sum_{j=0}^{n} a_{n-j} s^j} \tag{2.2}$$

式(2.2)称为系统的传递函数。

3. 权函数

若系统(初始条件为零)受一个理想脉冲函数 $\delta(t)$ 的作用，其响应为 $g(t)$，则 $g(t)$ 就称为该系统的权函数，或称脉冲过渡函数。理想的脉冲函数 $\delta(t)$ 的定义为

$$\begin{cases} \delta(t) = \begin{cases} \infty, t = 0 \\ 0, t \neq 0 \end{cases} \\ \int_0^\infty \delta(t)\mathrm{d}t = 1 \end{cases} \tag{2.3}$$

若在系统上施加一个任意作用函数 $u(t)$，则其响应 $y(t)$ 可通过以下卷积积分求出

$$y(t) = \int_0^\tau u(\tau) g(t-\tau)\mathrm{d}t \tag{2.4}$$

可以证明 $g(t)$ 与 $G(s)$ 构成一对拉氏变换对，即

$$\mathcal{L}[g(t)] = G(s) \tag{2.5}$$

4. 状态空间描述

以上三种模型都只描述了系统输入与输出之间的关系，而没有描述系统内部的情况，所以这些模型称为外部模型。从仿真的角度来看，为在计算机上对系统的模型进行试验，就要在计算机上复现(实现)这个模型。有时，仅仅实现系统输入与输出之间的关系是不够的，还必须复现模型的内部变量，即状态变量，因此仿真要求采用系统内部模型，即状态空间模型。

状态空间描述的一般形式为

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u \tag{2.6}$$

$$y = \boldsymbol{C}\boldsymbol{x} \tag{2.7}$$

式(2.6)称为状态方程，式(2.7)称为输出方程。其中 $\boldsymbol{A}$ 是 $n \times n$ 维矩阵，$\boldsymbol{B}$ 是 $n \times 1$ 维矩阵，$\boldsymbol{C}$ 是 $1 \times n$ 维矩阵。对形如式(2.1)的单输入单输出的 n 阶系统，引进 n 个内部状态变量 x_1，x_2，$\cdots$，x_n，易于将其转换为上述形式的状态方程。作用函数为单输入 u，输出变量为单输出 y。

仿真时，必须将系统的外部模型转换成内部模型，也就是建立与输入输出特性等价的状态方程。这个问题在控制理论中称为实现问题。在 2.2 节中我们将介绍几种模型结构变换方法。

2.1.2 离散时间模型

假定一个系统的输入量、输出量及其内部状态量是时间的离散函数，即为一时间序列：$\{u(kT)\}$，$\{y(kT)\}$，$\{x(kT)\}$，其中 T 为离散时间间隔(有时为简单起见，在序列中不写 T，而直接用$\{u(k)\}$，$\{y(k)\}$，$\{x(k)\}$来表示)，那么我们可以用离散时间模型来描述它。离散时间模型也有差分方程、z 传递函数、权序列、离散状态空间模型四种形式。

1. 差分方程

差分方程的一般表达形式为

$$a_0y(n+k)+a_1y(n+k-1)+\cdots+a_ny(k)=b_1u(n+k-1)+\cdots+b_nu(k) \quad (2.8)$$

不失一般性，可设 $a_0=1$。若引进后移算子 q^{-1}，$q^{-1}y(k)=y(k-1)$，则式(2.8)可改写成

$$\sum_{j=0}^{n}a_jq^{-j}y(n+k)=\sum_{j=1}^{n}b_jq^{-j}u(n+k)$$

即

$$\frac{y(n+k)}{u(n+k)}=\frac{\sum_{j=1}^{n}b_jq^{-j}}{\sum_{j=0}^{n}a_jq^{-j}} \quad 或 \quad \frac{y(k)}{u(k)}=\frac{\sum_{j=1}^{n}b_jq^{-j}}{\sum_{j=0}^{n}a_jq^{-j}} \quad (2.9)$$

2. z 传递函数

若系统的初始条件均为零，即 $y(k)=u(k)=0$，$(k\leqslant 0)$，对式(2.8)两边取 z 变换，则可得

$$(a_0+a_1z^{-1}+\cdots+a_nz^{-n})Y(z)=(b_1z^{-1}+\cdots+b_nz^{-n})U(z)$$

定义

$$G(z)=\frac{Y(z)}{U(z)}$$

$G(z)$称为系统的 z 传递函数，则

$$G(z)=\frac{\sum_{j=1}^{n}b_jz^{-j}}{\sum_{j=0}^{n}a_jz^{-j}} \quad (2.10)$$

可见，在系统初始条件均为零的情况下，z^{-1} 与 q^{-1} 等价。

3. 权序列

若对一初始条件均为零的系统施加一单位脉冲序列 $\delta(k)$，则其响应称为该系统的权序列$\{h(k)\}$，而 $\delta(k)$定义为

$$\delta(k)=\begin{cases}1, & k=0\\ 0, & k\neq 0\end{cases}$$

若输入序列为任意一个$\{u(k)\}$，则根据卷积定义，可得此时系统响应 $y(k)$为

$$y(k) = \sum_{i=0}^{k} u(i)h(k-i) \tag{2.11}$$

可以证明

$$\mathcal{Z}\{h(k)\} = H(z) \tag{2.12}$$

4. 离散状态空间模型

以上三种模型由于只描述了系统的输入序列与输出序列之间的关系因而称为外部模型。仿真要求采用内部模型，即离散状态空间模型。例如对式(2.8)所示之系统，若设

$$\sum_{j=0}^{n} a_j q^{-j} x(n+k) = u(k) \tag{2.13}$$

并令

$$q^{-j}x(n+k) = x_{n-j+1}(k) \qquad (j = 1,2,\cdots,n) \tag{2.14}$$

则有

$$\sum_{j=1}^{n} a_j q^{-j} x(n+k) + a_0 x(n+k) = u(k)$$

即

$$\sum_{j=1}^{n} a_j x_{n-j+1}(k) + a_0 x(n+k) = u(k)$$

设 $a_0=1$，并令 $x(n+k)=x_n(k+1)$，则不难得到

$$x(n+k) = x_n(k+1) = -\sum_{j=1}^{n} a_j x_{n-j+1}(k) + u(k) \tag{2.15}$$

根据式(2.14)及式(2.15)可列出以下 n 个一阶差分方程

$$\begin{aligned}
x_1(k+1) &= x_2(k)\\
x_2(k+1) &= x_3(k)\\
&\vdots\\
x_{n-1}(k+1) &= x_n(k)\\
x_n(k+1) &= -a_n x_1(k) - a_{n-1} x_2(k) - \cdots - a_1 x_n(k) + u(k)
\end{aligned}$$

写成矩阵形式

$$\boldsymbol{x}(k+1) = \boldsymbol{F}\boldsymbol{x}(k) + \boldsymbol{G}u(k) \tag{2.16}$$

其中

$$\boldsymbol{F} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ -a_n & -a_{n-1} & \cdots & \cdots & -a_1 \end{bmatrix}, \quad \boldsymbol{G} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

将式(2.13)代入式(2.8)，可得

$$\sum_{j=0}^{n} a_j q^{-j} y(k) = \sum_{j=1}^{n} b_j q^{-j} u(k) = \sum_{j=1}^{n} b_j q^{-j} \sum_{j=0}^{n} a_j q^{-j} x(n+k)$$

故有

$$y(k) = \sum_{j=1}^{n} b_j q^{-j} x(n+k) = \sum_{j=1}^{n} b_j x_{n-j+1}(k) = \boldsymbol{\Gamma}\boldsymbol{x}(k) \tag{2.17}$$

其中

$$\boldsymbol{\Gamma} = [b_n \quad b_{n-1} \quad \cdots \quad b_1]$$

式(2.16)及式(2.17)就称为系统离散状态空间模型。

2.1.3 连续-离散混合模型

假定有一系统,它的诸环节中有的环节的状态量是连续变量,而有的环节的状态量是离散变量。例如用数字计算机控制一连续对象而组成的计算机控制系统就属于这一类系统(如图 2.1 所示,图中 T 表示以 T 为周期的采样开关)。对于这类系统,它的离散部分(如图 2.1 中的数字计算机)可用离散时间模型来描述,而它的连续部分(如图 2.1 中的连续对象)则可用连续时间模型来描述。图中的保持器是一个将离散信号$\{u(kT)\}$恢复成连续信号的装置,它又应怎样来描述呢?这就要用到脉冲序列函数的概念。

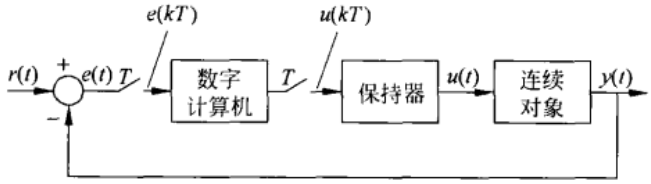


图 2.1 计算机控制系统

首先我们假设数字计算机完成的运算关系为 1,即

$$u(kT) = e(kT)$$

则图 2.1 中的数字计算机和保持器两部分可合并为图 2.2 的形式。

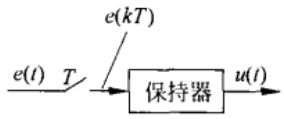


图 2.2 采样开关及保持器

设保持器为零阶,即它使离散信号$\{e(kT)\}$变成阶梯状分段信号 $u(t)$,即

$$u(t) = \sum_{k=0}^{\infty} e(kT)[1(t-kT) - 1(t-kT-T)]$$

对上式取拉氏变换可得

$$u(s) = \sum_{k=0}^{\infty} e(kT)\left[\frac{\mathrm{e}^{skT} - \mathrm{e}^{-s(k+1)T}}{s}\right] = \frac{1-\mathrm{e}^{sT}}{s}\sum_{k=0}^{\infty} e(kT)\mathrm{e}^{-skT} \tag{2.18}$$

令

$$G(s) = \frac{1-\mathrm{e}^{-sT}}{s}$$

$$T(s) = \sum_{k=0}^{\infty} e(kT)\mathrm{e}^{-skT}$$

则有

$$U(s) = G(s)T(s) \tag{2.19}$$

可以证明

$$T(s) = \mathcal{L}\left[\sum_{k=0}^{\infty} e(t)\delta(t-kT)\right] = \mathcal{L}[e^*(t)] \tag{2.20}$$

因此可以设想，$e^*(t)$是图 2.3 所示的虚拟系统所产生的脉冲列。图中的开关设想为每隔 T 秒闭合一次，以产生脉冲序列

$$e^*(t) = \sum_{k=0}^{\infty} e(t)\delta(t-KT)$$

同理，保持器可以定义为将 $e^*(t)$这样的脉冲列变成阶梯状波形的环节。综上所述，图 2.4(a)与(b)这两个系统是等价的。

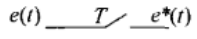


图 2.3 $e^*(t)$的等效

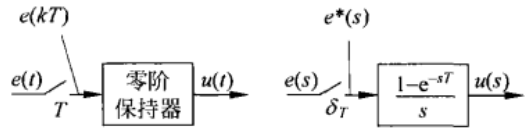


图 2.4 采样与保持器

因此，图 2.1 的计算机控制系统的数学模型如图 2.5 所示。其中 $D(z)$为 $e^*(t)$与 $u^*(t)$拉氏变换之比，并取 $z=e^{sT}$后所得的 z 传递函数。可以证明，它与先进行 $e(kT)$与 $u(kT)$的 z 变换，然后取两者之比是完全相同的。$G_h(s)$为保持器传递函数，$G(s)$为连续对象的传递函数。

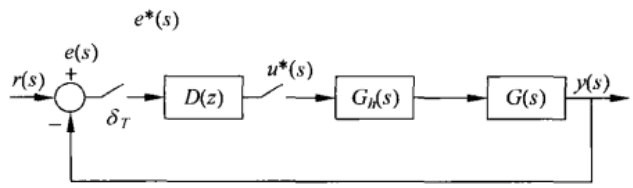


图 2.5 计算机控制系统的数学模型

2.2 连续系统模型设计

前面讨论的高阶微分方程、传递函数、权函数这三种模型描述系统的输入与输出的关系，称作系统的“外部模型”，而状态方程称为系统的“内部模型”。为了对一个连续系统进行仿真，首先要进行模型设计，任务之一就是要将这个系统的模型转换成计算机程序便于实现的形式。在下面的讨论中我们会发现，把系统的各种描述形式转换成内部模型—状态空间模型是最便于实现的。

2.2.1 外部模型到内部模型的变换

例如，有一个连续系统，它的数学模型如下式所示

$$\frac{d^n y}{dt^n} + a_1 \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_n y = u \tag{2.21}$$

今引进 n 个状态变量

$$x_1 = y$$

$$x_2 = \dot{x}_1 = \frac{\mathrm{d}y}{\mathrm{d}t}$$

$$x_3 = \dot{x}_2 = \frac{\mathrm{d}^2 y}{\mathrm{d}t^2}$$

$$\vdots$$

$$x_n = \dot{x}_{n-1} = \frac{\mathrm{d}^{n-1} y}{\mathrm{d}t^{n-1}}$$

则有

$$\dot{x}_n = \frac{\mathrm{d}^n y}{\mathrm{d}t^n} = -a_1 \frac{\mathrm{d}^{n-1} y}{\mathrm{d}t^{n-1}} - a_2 \frac{\mathrm{d}^{n-2} y}{\mathrm{d}t^{n-2}} - \cdots - a_n y + u$$

$$= -a_1 x_n - a_2 x_{n-1} - \cdots - a_n x_1 + u$$

将上述 n 个一阶微分方程写成矩阵形式可得

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} u \tag{2.22}$$

$$y = [1 \quad 0 \quad 0 \quad \cdots \quad 0]\boldsymbol{x} \tag{2.23}$$

将式(2.22)及式(2.23)用图形的方式表达出来，即为图 2.6。

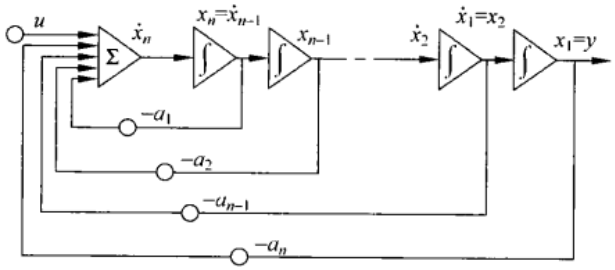


图 2.6 在模拟机上仿真的模型

若在模拟机上对式(2.22)及式(2.23)所示的系统进行仿真，那么只要对图 2.6 稍加处理(比如考虑时间比例尺、幅值比例尺、运算放大器的符号等)就可直接使用。所以这种图通常被称为模拟图，或称在模拟机上进行仿真的仿真模型。

若系统的微分方程如式(2.1)所示，即方程右端不仅有 $u(t)$，还有 $u(t)$的各阶导数及其初值，则上述方法难以直接使用，因为状态变量 $x_i(i=1,2,\cdots,n)$的初值与 $y^{(i)}(t_0)$，$u^{(i)}(t_0)$之间的关系不易得到。这种情况下的转换方法将在 2.2.3 节讨论。

由外部模型变换到内部模型不是唯一的，它可以写成多种不同的形式，所以仿真模型也并不是唯一的。需要指出的是，既然一个系统有多种实现，那么必有一部分的 $\boldsymbol{A}$ 是最小维数的，在控制理论中称为最小实现。由图 2.6 可知，$\boldsymbol{A}$ 的维数直接对应着仿真模型中的积分器的个数，比如 $\boldsymbol{A}$ 为 n 维，则仿真模型中就需要有 n 个积分器。显然，单纯从仿真模型的简单性方面来看，我们希望采用系统的最小实现作为仿真模型。

由给定的传递函数阵或脉冲过渡函数阵来建立与输入输出特性等价的状态方程，在控

制理论中把这类问题称作“实现问题”。也就是，给定一个传递函数阵 $\boldsymbol{G}(s)$，寻找一个状态空间方程$(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})$，使之成立

$$\boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}=\boldsymbol{G}(s)$$

那么，便称$(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})$是系统 $\boldsymbol{G}(s)$的实现。最小实现就是给定传递函数阵 $\boldsymbol{G}(s)$的假想结构的最简形式。最小实现的充分必要条件是$(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})$为完全能控且完全能观。

比如有一线性系统，它可用下面的微分方程来描述

$$\frac{\mathrm{d}^3y}{\mathrm{d}t^3}+6\frac{\mathrm{d}^2y}{\mathrm{d}t^2}+11\frac{\mathrm{d}y}{\mathrm{d}t}+6y=\frac{\mathrm{d}u}{\mathrm{d}t}+u \tag{2.24}$$

今已知 $y(t)$及 $u(t)$以及它们的各阶导数之初值均为零，所以式(2.24)也可转换成如下传递函数

$$G(s)=\frac{y(s)}{u(s)}=\frac{s+1}{s^3+6s^2+11s+6} \tag{2.25}$$

可将该系统的状态空间方程中的 $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}$ 写成

$$\boldsymbol{A}=\begin{bmatrix}0&1&0\\0&0&1\\-6&-11&-6\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}0\\0\\1\end{bmatrix},\quad \boldsymbol{C}=\begin{bmatrix}1&1&0\end{bmatrix} \tag{2.26}$$

因此仿真模型中要求有 3 个积分器。

利用控制理论中有关能控性与能观性的理论，可以判断上述状态空间方程并不是完全能观的，因此不是最小实现。

由式(2.25)可知，$G(s)$的分母可以分解为$(s+1)(s+2)(s+3)$，因此单纯从输入输出的观点来看，它与式(2.27)是等价的。

$$G(s)=\frac{1}{(s+2)(s+3)} \tag{2.27}$$

因此可得另一个新的 $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}$，如下所示

$$\boldsymbol{A}=\begin{bmatrix}0&1\\-6&-5\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}0\\1\end{bmatrix},\quad \boldsymbol{C}=\begin{bmatrix}1&0\end{bmatrix} \tag{2.28}$$

这个$(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})$是完全能控和能观的，因此是最小实现。利用这个模型来仿真只要有两个积分器就可以了。

对于多输入多输出的系统利用最小实现构造仿真模型有更加重要的意义，现举例说明。设系统的输入为 $u_1(t)$、$u_2(t)$，输出为 $y_1(t)$、$y_2(t)$，它们之间的关系可用传递函数阵来表示

$$\boldsymbol{G}(s)=\begin{bmatrix}\dfrac{2s+5}{s^2+5s+6}&\dfrac{2s^2+7s+7}{s^3+6s^2+11s+6}\\[2ex]\dfrac{3s^2+12s+11}{s^3+6s^2+11s+6}&\dfrac{s^2+6s+7}{s^2+5s+6}\end{bmatrix}$$

从表面上看，由于 $\boldsymbol{G}(s)$中的 $g_{11}(s)$和 $g_{22}(s)$为二阶，而 $g_{12}(s)$及 $g_{21}(s)$为三阶，因此似乎需要 10 个积分器才能构造出这个系统仿真模型。其实不然，为了用最小数量的积分器，可以设法求出 $\boldsymbol{G}(s)$的最小实现，先将 $\boldsymbol{G}(s)$中各分量分解，得

$$\boldsymbol{G}(s)=\begin{bmatrix}\dfrac{1}{s+2}+\dfrac{1}{s+3}&\dfrac{1}{s+1}-\dfrac{1}{s+2}+\dfrac{2}{s+3}\\[2ex]\dfrac{1}{s+1}+\dfrac{1}{s+2}+\dfrac{1}{s+3}&1-\dfrac{1}{s+2}+\dfrac{2}{s+3}\end{bmatrix}$$

显然该系统只有三种模态$\frac{1}{s+1},\frac{1}{s+2},\frac{1}{s+3}$。即$\boldsymbol{G}(s)$可改写为

$$\begin{aligned}\boldsymbol{G}(s)&=\begin{bmatrix}0&0\\0&1\end{bmatrix}+\begin{bmatrix}0&1\\1&0\end{bmatrix}\frac{1}{s+1}+\begin{bmatrix}1&-1\\1&-1\end{bmatrix}\frac{1}{s+2}+\begin{bmatrix}1&2\\1&2\end{bmatrix}\frac{1}{s+3}\\&=\begin{bmatrix}0&0\\0&1\end{bmatrix}+\begin{bmatrix}0&1\\1&0\end{bmatrix}\frac{1}{s+1}+\begin{bmatrix}1\\1\end{bmatrix}\frac{1}{s+2}\begin{bmatrix}1&-1\end{bmatrix}+\begin{bmatrix}1\\1\end{bmatrix}\frac{1}{s+3}\begin{bmatrix}1&2\end{bmatrix}\end{aligned}$$

因此有

$$\begin{aligned}\begin{bmatrix}Y_1(s)\\Y_2(s)\end{bmatrix}&=\begin{bmatrix}0&0\\0&1\end{bmatrix}\begin{bmatrix}U_1(s)\\U_2(s)\end{bmatrix}+\begin{bmatrix}0&1\\1&0\end{bmatrix}\frac{1}{s+1}\begin{bmatrix}U_1(s)\\U_2(s)\end{bmatrix}\\&+\begin{bmatrix}1\\1\end{bmatrix}\frac{1}{s+2}\begin{bmatrix}1&-1\end{bmatrix}\begin{bmatrix}U_1(s)\\U_2(s)\end{bmatrix}+\begin{bmatrix}1\\1\end{bmatrix}\frac{1}{s+3}\begin{bmatrix}1&2\end{bmatrix}\begin{bmatrix}U_1(s)\\U_2(s)\end{bmatrix}\end{aligned}$$

将上述方程画成模拟图，如图 2.7 所示。

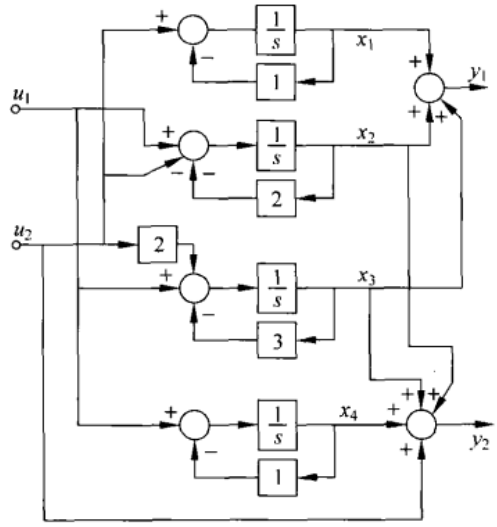


图 2.7 多输入多输出系统仿真模型

它们的状态空间模型为

$$\begin{bmatrix}\dot{x}_1\\\dot{x}_2\\\dot{x}_3\\\dot{x}_4\end{bmatrix}=\begin{bmatrix}-1&&&0\\&-2&&\\&&-3&\\0&&&-1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}+\begin{bmatrix}0&1\\1&-1\\1&2\\1&0\end{bmatrix}\begin{bmatrix}u_1\\u_2\end{bmatrix}$$

$$\begin{bmatrix}y_1\\y_2\end{bmatrix}=\begin{bmatrix}1&1&1&0\\0&1&1&1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}+\begin{bmatrix}0&0\\0&1\end{bmatrix}\begin{bmatrix}u_1\\u_2\end{bmatrix}$$

由此可见，该系统仅要求 4 个积分器即可在计算机上实现，可以证明，这是该系统的最小实现。

2.2.2 系统状态初始值变换

在2.2.1节中我们讨论了系统初始条件为零的情况，如果系统是非零初始条件，那么从外部模型变换到内部模型还必须考虑如何将给定的初始条件，通常是给定 $y(t)$，$u(t)$ 及其各阶导数的初始值，转变为相应的状态变量的初始值。

若系统是由如下一般形式的 n 阶微分方程来描述

$$a_0\frac{\mathrm{d}^n y}{\mathrm{d}t^n}+a_1\frac{\mathrm{d}^{n-1}y}{\mathrm{d}t^{n-1}}+\cdots+a_{n-1}\frac{\mathrm{d}y}{\mathrm{d}t}+a_n y=c_0\frac{\mathrm{d}^n u}{\mathrm{d}t^n}+c_1\frac{\mathrm{d}^{n-1}u}{\mathrm{d}t^{n-1}}+\cdots+c_{n-1}\frac{\mathrm{d}u}{\mathrm{d}t}+c_n u$$

系统初始条件为 $(y^{(i)}(t_0)=y_0^{(i)}, u^{(i)}(t_0)=u_0^{(i)} \quad i=0,1,2,\cdots,n-1)$。

现将其转换成 n 个一阶微分方程，该一阶微分方程组的状态变量记为 $x_j(j=1,2,\cdots,n)$，如果它们满足如下关系

$$x_1 = a_0 y - c_0 u \tag{2.29}$$

$$\vdots$$

$$\dot{x}_j = x_{j+1} - a_j y + c_j u \tag{2.30}$$

$$\vdots$$

$$\dot{x}_n = -a_n y + c_n u \tag{2.31}$$

$$y = \frac{1}{a_0}(x_1 + c_0 u) \tag{2.32}$$

则称该状态方程与原方程等价。这种变换称为伴随方程法，下面对此作简要说明。实际上，将式(2.29)两边分别进行微分 n 次，可得

$$p^n x_1 = a_0 p^n y - c_0 p^n u \tag{2.33}$$

其中 p 为微分算子符号。对式(2.30)两边分别进行 $n-j(j=1,2,\cdots,n-1)$ 次微分，可得

$$p^{n-j+1}x_j = p^{n-j}x_{j+1} - a_j p^{n-j} y + c_j p^{n-j} u \tag{2.34}$$

对式(2.31)也引入微分算子

$$p x_n = -a_n y + c_n u \tag{2.35}$$

将式(2.33)左右两边同乘以 -1 之后和式(2.34)、式(2.35)所包括的 n 个等式左右两边分别相加，消去同类项，稍加整理后就得到原高阶微分方程，表明两者之间的等价关系。

伴随方程法显式地表示了状态变量与原输入输出变量及其高阶导数之间的关系，因而易于进行初始值的转换。这样得到状态方程及输出方程

$$\begin{cases}\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u \\ y = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}u\end{cases} \tag{2.36}$$

其中

$$\boldsymbol{A}=\begin{bmatrix} -a_1 & 1 & 0 & \cdots & 0 \\ -a_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_{n-1} & 0 & 0 & \cdots & 1 \\ -a_n & 0 & 0 & \cdots & 0 \end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix} c_1 - c_0 a_1 \\ c_2 - c_0 a_2 \\ \vdots \\ c_n - c_0 a_n \end{bmatrix}$$

$$\boldsymbol{C}=\frac{1}{a_0}[1\quad 0\quad \cdots\quad 0],\quad \boldsymbol{D}=c_0/a_0$$

其初始值转换方程为(设 $a_0=1$)

$$\begin{bmatrix}x_{10}\\x_{20}\\\vdots\\x_{n0}\end{bmatrix}=\begin{bmatrix}1&0&\cdots&0\\a_1&1&&0\\\vdots&\vdots&\ddots&\vdots\\a_{n-1}&a_{n-2}&\cdots&1\end{bmatrix}\begin{bmatrix}y_0\\\dot{y}_0\\\vdots\\y_0^{(n-1)}\end{bmatrix}-\begin{bmatrix}c_0&0&\cdots&0\\c_1&c_0&&0\\\vdots&\vdots&\ddots&\vdots\\c_{n-1}&c_{n-2}&\cdots&c_0\end{bmatrix}\begin{bmatrix}u_0\\\dot{u}_0\\\vdots\\u_0^{(n-1)}\end{bmatrix}$$

伴随方程有多种形式,因而得到的状态方程也不唯一。那么,实现这种初始值转换的条件是什么呢?考虑转换后得到的系统状态空间模型为

$$\begin{cases}\dot{\boldsymbol{x}}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}u\\y=\boldsymbol{C}\boldsymbol{x}\end{cases}$$

即假定 u 的 n 阶导数项的系数 $c_0=0$,已知系统的初始条件为

$$y(0),\dot{y}(0),\cdots,y^{(n-1)}(0)$$

$$u(0),\dot{u}(0),\cdots,u^{(n-2)}(0)$$

则为了由上述初始值求出状态变量 $x_1,x_2,\cdots,x_n$ 的初始值,可列出以下方程

$$y(t)=\boldsymbol{C}\boldsymbol{x}(t)$$

$$\dot{y}(t)=\boldsymbol{C}\dot{\boldsymbol{x}}(t)=\boldsymbol{C}\boldsymbol{A}\boldsymbol{x}(t)+\boldsymbol{C}\boldsymbol{B}u(t)$$

$$\ddot{y}=\boldsymbol{C}\ddot{\boldsymbol{x}}(t)=\boldsymbol{C}\boldsymbol{A}\dot{\boldsymbol{x}}(t)+\boldsymbol{C}\boldsymbol{B}\dot{u}(t)=\boldsymbol{C}\boldsymbol{A}^2\boldsymbol{x}(t)+\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}u(t)+\boldsymbol{C}\boldsymbol{B}\dot{u}(t)$$

$$\vdots$$

于是,可得矩阵方程

$$\boldsymbol{y}(t)=\boldsymbol{\theta}\boldsymbol{x}(t)+\boldsymbol{T}\boldsymbol{u}(t)\tag{2.37}$$

其中

$$\boldsymbol{y}(t)=[y(t)\quad\dot{y}(t)\quad\cdots\quad y^{(n-1)}(t)]^{\mathrm{T}}$$

$$\boldsymbol{u}(t)=[u(t)\quad\dot{u}(t)\quad\cdots\quad u^{n-1}(t)]^{\mathrm{T}}$$

$$\boldsymbol{\theta}=\begin{bmatrix}\boldsymbol{C}\\\boldsymbol{C}\boldsymbol{A}\\\vdots\\\boldsymbol{C}\boldsymbol{A}^{n-1}\end{bmatrix},\quad \boldsymbol{T}=\begin{bmatrix}0&0&\cdots&0\\\boldsymbol{C}\boldsymbol{B}&0&\cdots&0\\\boldsymbol{C}\boldsymbol{A}\boldsymbol{B}&\boldsymbol{C}\boldsymbol{B}&&\\\vdots&\vdots&&\vdots\\\boldsymbol{C}\boldsymbol{A}^{n-2}\boldsymbol{B}&\boldsymbol{C}\boldsymbol{A}^{n-3}\boldsymbol{B}&\cdots&\boldsymbol{C}\boldsymbol{B}\quad 0\end{bmatrix}$$

均为 $n\times n$ 方阵。

由式(2.37)可得

$$\boldsymbol{x}(t)=\boldsymbol{\theta}^{-1}[\boldsymbol{y}(t)-\boldsymbol{T}\boldsymbol{u}(t)]\tag{2.38}$$

即若$\boldsymbol{\theta}^{-1}$存在,则可由式(2.38)求出 $\boldsymbol{x}(t)$的初始值。由控制理论可知,$\boldsymbol{\theta}$是$(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})$的能观判别阵,若$(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})$是完全能观的,则$\boldsymbol{\theta}$非奇异。这就是说,由高阶微分方程输入输出变量初始值转变为状态初始值的条件是:内部模型$(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})$是完全能观的。

例 2.1 设有一个系统,它的微分方程为

$$\ddot{y}+3\dot{y}+2y=\dot{u}+u$$

已知系统的初始条件为

$$\dot{y}(0)=1,\quad y(0)=1,\quad u(0)=0$$

试确定该系统的内部模型，并给出状态变量的初值。

解：先按式(2.26)给出系统的能控标准型

$$A=\begin{bmatrix}0 & 1\\ -2 & -3\end{bmatrix},\quad B=\begin{bmatrix}0\\1\end{bmatrix},\quad C=[1\quad 1]$$

它的$\boldsymbol{\theta}=\begin{bmatrix}1 & 1\\ -2 & -2\end{bmatrix}$，这是一个奇异矩阵，没有逆阵，因此是不完全能观的，无法求出状态变量的初值。为求出状态变量的初值，必须选 x_1,x_2 使$(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})$为完全能观的。今令

$$x_1=y$$

$$x_2=\dot{y}+3y-u=\dot{x}_1+3x_1-u$$

则有

$$\dot{x}_1=x_2-3x_1+u$$

$$\dot{x}_2=\ddot{y}+3\dot{y}-\dot{u}=-2y+u=-2x_1+u$$

故可得下列状态空间模型

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}-3 & 1\\ -2 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}1\\1\end{bmatrix}u$$

$$y=[1\quad 0]\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

x_1,x_2 的初值为

$$\begin{bmatrix}x_1(0)\\ x_2(0)\end{bmatrix}=\begin{bmatrix}y(0)\\ \dot{y}(0)+3y(0)-u(0)\end{bmatrix}=\begin{bmatrix}1\\4\end{bmatrix}$$

■

2.2.3 面向结构图的模型变换

工程上常常将系统描述为结构图的形式，因此，工程技术人员更习惯面向结构图。仿真模型设计的任务就是通过计算机自动将结构图模型转换成一阶微分方程组，以便仿真程序的实现，同时要建立结构图的变量与内部模型变量及参数的映射关系。本节讨论不包括非线性环节的结构图的转换，至于包括非线性环节的结构图的变换方法将在以后章节中讨论。

1. 典型环节的选择

结构图是由各种典型环节构成的。通常一个系统中比较常见的典型环节如下：

积分环节$\dfrac{K}{s}$　　比例积分环节 $K_1+\dfrac{K_2}{s}=\dfrac{K_1s+K_2}{s}$

惯性环节$\dfrac{K}{Ts+1}$　　一阶超前(或滞后)环节 $K\dfrac{T_1s+1}{T_2s+1}$

二阶振荡环节$\dfrac{K}{T^2s^2+2\zeta Ts+1}$

选择什么样的环节作为模型描述的基础，是模型结构变换十分重要的一步。选择典型

环节的原则是要有典型性，即由它可组成各种动态环节，组成系统简便，由计算机将它转换成系统的微分方程组容易实现。

选用图 2.8 所示的一阶领先(滞后)环节来作为典型环节是一种比较有效的方法，因为这种典型环节可以很容易地表示上述$\frac{K}{s}$，$\frac{K}{Ts+1}$，$K_1+\frac{K_2}{s}$，$K\frac{T_1s+1}{T_2s+1}$等常见的环节，至于二阶振荡环节则只要用两个这样的典型环节串联，并加上一个负反馈即可得到。

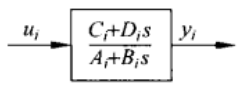


图 2.8 一阶领先(滞后)环节

下面我们来推导一下如何用程序来确定由这样的典型环节构成的系统状态方程。由图 2.8 可知，每个环节可写成如下的微分方程

$$B_i\frac{\mathrm{d}y_i}{\mathrm{d}t}+A_iy_i=D_i\frac{\mathrm{d}u_i}{\mathrm{d}t}+C_iu_i \tag{2.39}$$

写成矩阵形式，则有

$$\boldsymbol{B}\dot{\boldsymbol{y}}+\boldsymbol{A}\boldsymbol{y}=\boldsymbol{D}\dot{\boldsymbol{u}}+\boldsymbol{C}\boldsymbol{u} \tag{2.40}$$

其中

$$\boldsymbol{A}=\begin{bmatrix}A_1 & & 0\\ & A_2 & \\ & \ddots & \\ 0 & & A_n\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}B_1 & & 0\\ & B_2 & \\ & \ddots & \\ 0 & & B_n\end{bmatrix}$$

$$\boldsymbol{C}=\begin{bmatrix}C_1 & & 0\\ & C_2 & \\ & \ddots & \\ 0 & & C_n\end{bmatrix},\quad \boldsymbol{D}=\begin{bmatrix}D_1 & & 0\\ & D_2 & \\ & \ddots & \\ 0 & & D_n\end{bmatrix}$$

2. 面向结构图系统方程描述

假定有一个如图 2.9 所示的系统，其中用方框框起来并标上 1,2,…符号的都是典型环节，比如第一个环节为$\frac{C_1+D_1s}{A_1+B_1s}$，第 i 个环节为$\frac{C_i+D_is}{A_i+B_is}$，而 α_2，α_4，β_4 等都是连接系数，每个环节有一个输入信号 u_i，有一个输出信号 y_i，对于每一个环节它们的方程是

$$y_i=\frac{C_i+D_is}{A_i+B_is}u_i\quad (i=1,2,3,4,5)$$

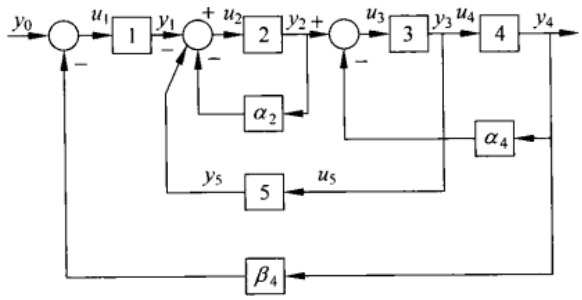


图 2.9 被仿真的系统框图

我们可以写出系统的动态方程为

$$(\boldsymbol{A}+\boldsymbol{B}s)\boldsymbol{y}=(\boldsymbol{C}+\boldsymbol{D}s)\boldsymbol{u} \tag{2.41}$$

$$\boldsymbol{u}=\boldsymbol{W}\boldsymbol{y}+\boldsymbol{W}_0 y_0 \tag{2.42}$$

在方程式(2.41)中，$\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D}$ 阵中的元素反映各环节的参数。该方程描述了各环节的输入、输出及参数之间的关系，各矩阵结构如下

$$\boldsymbol{A}=\begin{bmatrix} A_1 & 0 & 0 & 0 & 0 \\ 0 & A_2 & 0 & 0 & 0 \\ 0 & 0 & A_3 & 0 & 0 \\ 0 & 0 & 0 & A_4 & 0 \\ 0 & 0 & 0 & 0 & A_5 \end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix} B_1 & 0 & 0 & 0 & 0 \\ 0 & B_2 & 0 & 0 & 0 \\ 0 & 0 & B_3 & 0 & 0 \\ 0 & 0 & 0 & B_4 & 0 \\ 0 & 0 & 0 & 0 & B_5 \end{bmatrix}$$

$$\boldsymbol{C}=\begin{bmatrix} C_1 & 0 & 0 & 0 & 0 \\ 0 & C_2 & 0 & 0 & 0 \\ 0 & 0 & C_3 & 0 & 0 \\ 0 & 0 & 0 & C_4 & 0 \\ 0 & 0 & 0 & 0 & C_5 \end{bmatrix},\quad \boldsymbol{D}=\begin{bmatrix} D_1 & 0 & 0 & 0 & 0 \\ 0 & D_2 & 0 & 0 & 0 \\ 0 & 0 & D_3 & 0 & 0 \\ 0 & 0 & 0 & D_4 & 0 \\ 0 & 0 & 0 & 0 & D_5 \end{bmatrix}$$

方程式(2.42)中 $\boldsymbol{W}$ 阵和 $\boldsymbol{W}_0$ 阵均为系统的连接矩阵，描述了系统各环节之间的连接关系。$\boldsymbol{W}$ 称为系统的内部连接矩阵，它描述了系统内部各环节的连接情况，$\boldsymbol{W}_0$ 称为外部输入的连接矩阵，它描述了外部输入对系统的作用情况。一个 n 阶系统的内部连接矩阵 $\boldsymbol{W}$ 是一个 $n\times n$ 的方阵，它的每个元素 W_{ij} 表示第 j 个环节的输出到第 i 个环节的输入之间的连接系数。例如，在图 2.9 中，$W_{14}=-\beta_4$，表示第 4 个环节对第一个环节的连接系数为 $-\beta_4$，而 $W_{11}=W_{12}=W_{13}=W_{15}=0$，表示 1,2,3,5 这 4 个环节对第 1 个环节无连接关系。

$\boldsymbol{W}_0$ 是描述外部输入信号对系统的连接情况，此例为单输入系统，故 $\boldsymbol{W}_0$ 是一个列矢量(若为多输入系统则 $\boldsymbol{W}_0$ 也是一个矩阵，它的列数等于输入量的个数)。W_{0j} 表示外部输入信号 y_0 作用在第 j 个环节上的作用系数，在图 2.9 中，y_0 只作用在第一个环节上，故 $W_{01}=1$，其它均为 0。

可以将方程式(2.42)展开

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix}=\begin{bmatrix} 0 & 0 & 0 & -\beta_1 & 0 \\ 1 & -\alpha_2 & 0 & 0 & -1 \\ 0 & 1 & 0 & -\alpha_4 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix}+\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} y_0$$

3. 系统方程转换

系统方程转换是指将结构图如何转换成一组一阶线性微分方程。将式(2.42)代入式(2.41)，则可得

$$\boldsymbol{B}\dot{\boldsymbol{y}}+\boldsymbol{A}\boldsymbol{y}=\boldsymbol{D}\boldsymbol{W}\dot{\boldsymbol{y}}+\boldsymbol{C}\boldsymbol{W}\boldsymbol{y}+\boldsymbol{D}\boldsymbol{W}_0\dot{y}_0+\boldsymbol{C}\boldsymbol{W}_0 y_0 \tag{2.43}$$

稍加整理后，式(2.43)方程可写成

$$\boldsymbol{Q}\dot{\boldsymbol{y}}=\boldsymbol{P}\boldsymbol{y}+\boldsymbol{V}_1 y_0+\boldsymbol{V}_2\dot{y}_0 \tag{2.44}$$

其中

$$\boldsymbol{Q}=\boldsymbol{B}-\boldsymbol{DW},\quad \boldsymbol{P}=\boldsymbol{CW}-\boldsymbol{A},\quad \boldsymbol{V}_1=\boldsymbol{CW}_0,\quad \boldsymbol{V}_2=\boldsymbol{DW}_0$$

如果 $\boldsymbol{Q}$ 阵的逆存在，那么对式(2.44)两边左乘 $\boldsymbol{Q}^{-1}$，则得

$$\dot{\boldsymbol{y}}=\boldsymbol{Q}^{-1}\boldsymbol{Py}+\boldsymbol{Q}^{-1}\boldsymbol{V}_1 y_0+\boldsymbol{Q}^{-1}\boldsymbol{V}_2\,\dot{y}_0 \tag{2.45}$$

这是一个标准的一阶常微分方程组。

关于式(2.45)有两点要特别加以说明如下：

(1) 输入函数导数项处理

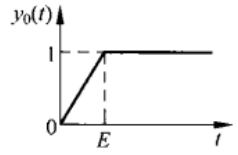


图 2.10 外加作用信号

由式(2.45)可知，在这个矩阵方程的右端有两项与外加作用信号 y_0 有关，一项是 $\boldsymbol{Q}^{-1}\boldsymbol{V}_1 y_0$，另一项是 $\boldsymbol{Q}^{-1}\boldsymbol{V}_2\,\dot{y}_0$。也就是说，为了计算右端函数，不仅要知道外加作用信号 y_0 本身，还要知道它的导数 $\dot{y}_0$。若外加作用函数是如图 2.10 所示的线性渐增函数，则有

$$y_0=\begin{cases}0, & t<0\\ t/E, & 0\leqslant t<E\\ 1, & t\geqslant E\end{cases}$$

同时有

$$\dot{y}_0=\begin{cases}0, & t<0\\ 1/E, & 0\leqslant t<E\\ 0, & t\geqslant E\end{cases}$$

当 $E\to 0$ 时，它就是单位阶跃阵，此时 $1/E\to\infty$，那么式(2.44)中右端的第三项将趋于无穷大。为了便于式(2.44)的计算，就要求 V_2 是零向量，这就是说，如果外加作用信号是阶跃信号，那么必须限制外加作用信号所用的那个环节 $D_i=0$。

(2) 保证 $\boldsymbol{Q}$ 阵逆存在

由上所述，只有当 $\boldsymbol{Q}$ 阵能求逆时，才能获得式(2.45)。什么情况下能保证 $\boldsymbol{Q}$ 阵有逆存在呢？从物理意义上讲，当系统中各环节不存在纯微分环节和(或)纯比例环节时就能保证 $\boldsymbol{Q}$ 阵可以求逆。

矩阵逆存在的充分必要条件是矩阵的行列式 $|\boldsymbol{Q}|$ 不等于 0。如果 $\boldsymbol{Q}$ 阵中某一列(行)的元素全为零或某些行或列线性相关，那么这个矩阵将不存在逆。例如当被仿真的系统中有纯比例或纯微分环节时就有可能发生这种情况。

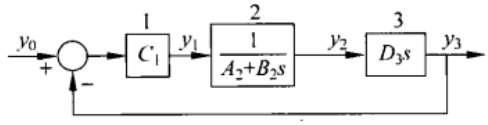


图 2.11 结构变换举例

现以图 2.11 为例加以说明。根据前面所述的方法，不难得到

$$\boldsymbol{A}=\begin{bmatrix}1 & 0 & 0\\ 0 & A_2 & 0\\ 0 & 0 & 1\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}0 & 0 & 0\\ 0 & B_2 & 0\\ 0 & 0 & 0\end{bmatrix},\quad \boldsymbol{C}=\begin{bmatrix}C_1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}$$

$$\boldsymbol{D}=\begin{bmatrix}0&0&0\\0&0&0\\0&0&D_3\end{bmatrix},\quad \boldsymbol{W}=\begin{bmatrix}0&0&-1\\1&0&0\\0&1&0\end{bmatrix},\quad \boldsymbol{W}_0=\begin{bmatrix}1\\0\\0\end{bmatrix}$$

则

$$\boldsymbol{Q}=\boldsymbol{B}-\boldsymbol{DW}=\begin{bmatrix}0&0&0\\0&B_2&0\\0&-D_3&0\end{bmatrix},\quad \boldsymbol{P}=\boldsymbol{CW}-\boldsymbol{A}=\begin{bmatrix}-1&0&-C_1\\1&-A_2&0\\0&0&-1\end{bmatrix},\quad \boldsymbol{V}_1=\boldsymbol{CW}_0=\begin{bmatrix}C_1\\0\\0\end{bmatrix}$$

因为 $\boldsymbol{Q}$ 阵中出现 1,3 两列全零，所以 $\boldsymbol{Q}^{-1}$ 不存在。其原因是 1,3 两个环节是比例和微分环节，它们都不存在相应 $\dot{y}_i(i=1,3)$。对于上述系统，可以将其结构加以变换，如图 2.12 所示。

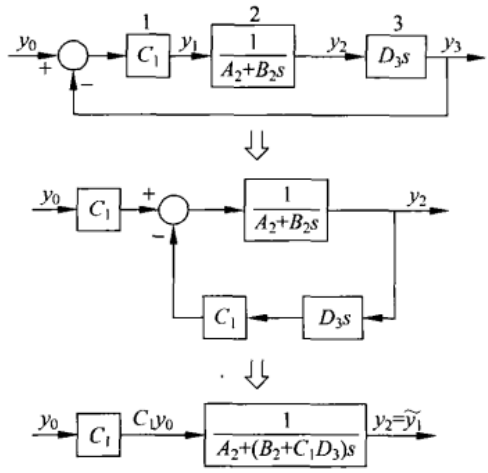


图 2.12 结构变换过程

显然，经过结构变换后，上述系统可用一个典型环节表示，这时的 $\tilde{\boldsymbol{Q}}$ 阵是有逆的。求出 y_2（即新系统的 $\tilde{y}_1$）后，还要由 $y_3=(sy_2)D_3$ 去算出实际系统的输出 y_3。

工业过程控制系统往往含有比例、微分或比例-微分环节。如果这些变换都要由人手工去完成实在是一件麻烦的事。下面介绍一种系统结构变换的程序法，以便通过计算机程序来自动完成。

假定整个系统的状态方程为

$$\boldsymbol{Q}\dot{\boldsymbol{y}}=\boldsymbol{P}\boldsymbol{y}+\boldsymbol{V}_1y_0 \tag{2.46}$$

$\boldsymbol{Q}$ 阵中有 $(N-M)$ 列元素为全零，这说明有 $(N-M)$ 个环节的 $\dot{\boldsymbol{y}}$ 不出现在方程的左端。也就是说，系统中有 $(N-M)$ 个代数方程。

系统结构变换程序法就是通过矩阵的初等变换，把系统中 $(N-M)$ 个代数方程分离出来。具体做法，就是对 $\boldsymbol{Q},\boldsymbol{P},\boldsymbol{V}_1$ 各矩阵线性变换，使上述方程中的各矩阵变为 $\tilde{\boldsymbol{Q}},\tilde{\boldsymbol{P}},\tilde{\boldsymbol{V}}_1$，从而 $\boldsymbol{y}$ 变为 $\tilde{\boldsymbol{y}}$，方程变为

$$\tilde{\boldsymbol{Q}}\,\dot{\tilde{\boldsymbol{y}}}=\tilde{\boldsymbol{P}}\,\tilde{\boldsymbol{y}}+\tilde{\boldsymbol{V}}_1y_0 \tag{2.47}$$

其中

$$\widetilde{\boldsymbol{Q}}=\left[\begin{array}{ccc:c}\widetilde{Q}_{11} & \cdots & \widetilde{Q}_{1M} & \\ \widetilde{Q}_{21} & \cdots & \widetilde{Q}_{2M} & \\ \vdots & & \vdots & 0 \\ \widetilde{Q}_{M1} & \cdots & \widetilde{Q}_{MM} & \\ \hdashline \widetilde{Q}_{M+1,1} & \cdots & \widetilde{Q}_{M+1,M} & \\ \vdots & & \vdots & 0 \\ \widetilde{Q}_{N,1} & \cdots & \widetilde{Q}_{N,M} & \end{array}\right]=\left[\begin{array}{c:c}\widetilde{Q}_M & 0 \\ \hdashline \widetilde{Q}_{N-M} & 0\end{array}\right] \tag{2.48}$$

$$\widetilde{\boldsymbol{P}}=\left[\begin{array}{ccc:cc}\widetilde{P}_{11} & \cdots & \widetilde{P}_{1,M} & & \\ \vdots & & \vdots & & 0 \\ \widetilde{P}_{M,1} & \cdots & \widetilde{P}_{M,M} & & \\ \hdashline \widetilde{P}_{M+1,1} & \cdots & \widetilde{P}_{M+1,M} & 0 & 1 \\ \vdots & & \vdots & \iddots & \\ \widetilde{P}_{N,1} & \cdots & \widetilde{P}_{N,M} & 1 & 0\end{array}\right]=\left[\begin{array}{c:cc} & 0 & \\ \widetilde{P}_M & & \\ \hdashline \widetilde{P}_{N-M} & 0 & 1 \\ & \iddots & \\ & 1 & 0\end{array}\right] \tag{2.49}$$

$$\widetilde{\boldsymbol{V}}_1=\left[\begin{array}{c}\widetilde{\boldsymbol{V}}_{1M} \\ \hdashline \widetilde{\boldsymbol{V}}_{1,N-M}\end{array}\right] \tag{2.50}$$

在$(N-M)$个环节变换的过程中，对 $\boldsymbol{Q},\boldsymbol{P},\boldsymbol{V}_1$ 作列变换，将使 $\boldsymbol{y}$ 原来的编号改变。因此$\tilde{\boldsymbol{y}}$是原来 $\boldsymbol{y}$ 进行重新编号的结果。所以程序中应将列变换的情况记录下来，以便在最后求出$\tilde{\boldsymbol{y}}$后，经过反变换而仍能恢复 $\boldsymbol{y}$。

完成变换后，系统可以分为 M 个微分方程与$(N-M)$个代数方程，而该 M 个微分方程的$\widetilde{\boldsymbol{Q}}_M$ 的逆存在。因此

$$\dot{\tilde{\boldsymbol{y}}}_M=\widetilde{\boldsymbol{Q}}_M^{-1}\widetilde{\boldsymbol{P}}_M\tilde{\boldsymbol{y}}_M+\widetilde{\boldsymbol{Q}}_M^{-1}\widetilde{\boldsymbol{V}}_{1M}y_0 \tag{2.51}$$

由此解得$\tilde{y}_1,\tilde{y}_2,\cdots,\tilde{y}_M$ 及其导数$\dot{\tilde{y}}_1,\dot{\tilde{y}}_2,\cdots,\dot{\tilde{y}}_M$，然后再解$(N-M)$个代数方程

$$\begin{aligned}&\sum_{j=1}^{M}\widetilde{Q}_{ij}\dot{\tilde{y}}_j=\sum_{j=1}^{N}\widetilde{P}_{ij}\tilde{y}_j+\widetilde{V}_{1i}y_0 \\ &i=M+1,M+2,\cdots,N\end{aligned} \tag{2.52}$$

由于当$(j=M+1,i=N)$；$(j=M+2,i=N-1)$；…；$(j=N,i=M+1)$时，$\widetilde{P}_{ij}=1$，因此方程式(2.52)可以写成

$$\tilde{y}_{N+M+1-i}=\sum_{j=1}^{M}\widetilde{Q}_{ij}\dot{\tilde{y}}_j-\sum_{j=1}^{M}\widetilde{P}_{ij}\tilde{y}_j-\widetilde{V}_{1i}y_0\quad(i=M+1,M+2,\cdots,N) \tag{2.53}$$

2.3 计算机辅助模型设计

经典的模型设计思路是：首先界定研究对象的边界和建模目标，利用已有的建模工具和成果，建立相应的数学模型。然而，随着研究对象的日益复杂，很难给出对象的数学描述。例如，有成千上万个电阻、电容、电感及各种类型门电路、放大电路组成的电路系统，尽管我们对每个元器件的模型描述均可给出，但很难直接给出其电路方程。又如，由众多零部件组成的机械产品，虽然我们可以方便地描述各个零部件的运动学及动力学关系，但是却由于各个零部件间存在多种复杂的力学耦合关系，因而也很难得到整个产品的运动学、动力学模型。计算机辅助模型设计就是为用户提供一种交互式建模环境，其目标是，在计算机支持下，实现模型的存储、综合，使系统建模过程进行得更加有效。

2.3.1 基于计算机的建模

人们在长期的科学研究中，已经对客观世界进行了大量的信息收集，得到了许多关于研究对象的知识和规律，不同的学科、不同的人基于不同的研究目标对研究对象已经建立了大量的模型。如何充分发挥计算机大容量的数据存储、高速的计算、准确的逻辑判断等能力，实现模型的综合利用，是仿真模型设计的重要内容。

一般说来，传统的建模手段依赖于专家的专业知识和技巧，建模对象的规模和复杂程度也必然受到限制。例如，人们可以对几十个到上百个电路元件组成的电路系统列出其电路方程，人工对电路方程进行处理，从而得到该电路系统的模型，然后由计算机计算。但是，当电路系统有成千上万个元件，特别是还包括许多非线性元件时，用人工方法就很难建立系统的模型。

基于计算机的建模就是要建立支持建模与仿真的计算机环境，实现对已知事实、局部知识、已有模型的处理，以达到对局部模型的集成与综合利用。

系统由实体或场构成，而实体或场用变量（属性）描述，对于已经认识的真实世界的模型，可以定义其输入变量、输出变量和状态变量，一般称其为静态结构。实体之间的相互作用或活动反映了系统的动态行为，称为动态结构。动态结构包含时间基，即系统的动态行为随时间变化的规律。基于计算机的建模方法将静态结构与动态结构分别建模，并能实现两者之间的集成与映射。

基于计算机的建模首先要定义实体库。实体的种类决定了建模的能力，任何一次仿真模型设计都是从实体库的元素中根据仿真对象选择一个子集，形成一个实例。实体的属性用变量描述，它只能包含从属于该类实体的变量，因此要基于实体库建立起相应的变量库。

模型的静态结构是指一个实体可以表现为一个或多个其它实体的组件，这就需要定义实体的结构，确定构造规则。既可以从基本实体（不可再分的实体）按规则组合成一个新实体，并实现其变量的映射与聚合关联；也可以从上而下进行定义，即先定义上层实体，然后按规则进行分解，形成下层的子实体集，并实现其变量的映射与关联分解。

模型的动态结构包括两个方面：实体的行为规律及相互作用规律。实体的行为规律的

定义要遵照物理、化学、生物、社会、经济等固有规律，采用适当的形式加以描述。这是人类对客观世界认识的已有知识的总结，要根据仿真的不同目标要求，提供多种描述形式。实体的相互作用规律本质上是实体行为规律的一部分，它重点描述实体之间的相互联系、相互作用。例如两个电路，在定义了两者之间的连接关系（静态结构）后，其相互作用的规律随之就确定了（电压关系、电流关系等），两个刚体若用铰连接则其相互之间的作用也就确定了，只是要用某种形式将其加以描述，以便最终能由计算机正确识别并正确加以处理。

模型的静态结构与动态结构均是基于数据库进行管理，从而形成模型库。Zeigler 称这种基于模型库的建模结构为操作式结构（operations structure）。他用掷币过程的模型设计为例加以说明（参见图 2.13），基于模型库建立了面向不同仿真目标的模型，其中连接模型的关系有两种：实线表示“组合—分解”，虚线表示“简化—细化”。例如，M_{13} 由 M_{11} 和 M_{12} 组成，而 M_{11} 和 M_{12} 分别是 M_5 和 M_9 的细化；M_{13} 是 M_{10} 的细化；M_{10} 由 M_3、M_6 和 M_7 组成，等等，这些关系可能是一个交错性的网状结构。如何保证这种网状式模型设计的有效性和正确性，是基于结构化建模的关键。

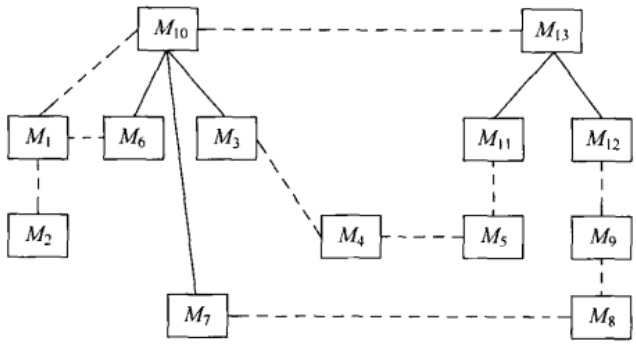


图 2.13 掷币过程的操作结构模型

对于这种操作式结构，用户在进行模型设计时，总有其自己的目的，因而也是从用户的角度来设计模型。但对任何一个可用的计算机建模支撑环境，其能力总是有限的，因此往往会给出一个具有一定通用度的设计框架，也称为域框架（scope frame）。域框架 E_M 可以被认为是最一般性的框架，它给出了基于某一模型库进行模型设计的最广泛的集合。如果某一模型 M 是从 E_M 导出的，则该模型 M 是可用的。

模型库应该是可扩充的，它不但可以保存已经得到验证过的模型，而且可以按照规则不断地添加新的模型。这些新的模型可以和已有的模型一样进行“组合—分解”及“简化—细化”等操作。

近些年来，元建模（meta modeling）技术得到许多领域的重视，因而基于元模型的计算机建模得到很大的发展。元模型就是关于模型的模型，是关于如何建立模型、模型的语义或模型之间如何集成和互操作等信息的描述。元模型由于比模型的抽象程度高，因此能够较好地解决模型集成中的问题。

一个模型的元模型是关于模型中的基本概念、基本关系、基本约束的语义，此模型的建模步骤、建模时应该遵循的原则和（或）此模型的局限性等隐含信息的显式描述，从而使模型的建立和使用有理可依。面向模型集成目的的元模型还要增加各个组成模型的元模型之间的关联或映射规则等模型间的信息。

模型是元模型的实例，元模型是模型的抽象。正如模型是对客观世界的抽象，元模型是对模型有关信息的概括和抽象，因此它的抽象层次更高。元模型的类型系统(ontology)比模型的类型系统更细致地划分了模型的类型系统中的各种概念、关系、约束等，因此它表达客观世界的能力更强。元模型的类型系统中通常还或多或少地包含有关于模型的控制和指导等信息的概括与抽象，将这些基本概念实例化就可以对某个具体模型的建模进行监控。

2.3.2 解释结构建模

大型或复杂系统由大量实体(可以是物理元件，也可以是装置，还可以是其它对象以及子系统)所组成。这些实体之间存在着错综复杂的相互作用关系，这些关系包括因果、顺序等逻辑关系、信息传递关系、数学函数关系等。结构建模的一个重要任务就是要能方便、灵活地正确定义实体之间的关系，图论为结构建模提供了有效工具。

作为实体集合的系统模型，用集合 $\boldsymbol{S}=\{s_1,s_2,\cdots,s_n\}$ 表示实体集，$s_i(i=1,2,\cdots,n)$ 是实体集合中的元素，$\boldsymbol{R}=\{(x,y)|\boldsymbol{W}(x,y)\}$ 表示在某种关系 $\boldsymbol{W}$ 下各实体间的关系值(是否存在关系 $\boldsymbol{W}$ 可用 0 或 1 表示)。这样集合 $\boldsymbol{S}$ 及其关系 $\boldsymbol{R}$ 就可表示系统中实体的结构模型，可用有向连接图或矩阵来描述。

用节点表示 $\boldsymbol{S}$ 中的元素，用节点间的有向线段(弧)表示关系 $\boldsymbol{R}$，并约定，如果 $(s_i,s_j)\in\boldsymbol{R}$，则在节点 s_i 和 s_j 之间存在一条弧，其方向由 s_i 指向 s_j。这样结构模型 $\{\boldsymbol{S},\boldsymbol{R}\}$ 就可用有向图 $\boldsymbol{G}=\{\boldsymbol{S},\boldsymbol{R}\}$ 来唯一地表示出来，其中 $\boldsymbol{S}$ 为节点集合，$\boldsymbol{R}$ 为有向弧集合。例如一个有 4 个节点的模型 $\{\boldsymbol{S},\boldsymbol{R}\}$，其中 $\boldsymbol{R}=\{[s_1,s_3],[s_2,s_1],[s_3,s_2],[s_2,s_4],[s_4,s_3]\}$，其对应的有向图如图 2.14 所示。

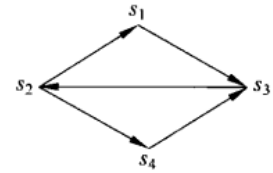


图 2.14 有向连接图

为了便于计算机存储，可以将有向连接图转换为连接矩阵，正如我们在 2.2.3 节中看到的那样。为了更具一般性，这里给出关联矩阵的概念。结构模型 $\{\boldsymbol{S},\boldsymbol{R}\}$ 的关联矩阵 $\boldsymbol{A}$ 定义为：设系统实体集合为 $\boldsymbol{S}=\{s_1,s_2,\cdots,s_n\}$，则 $n\times n$ 矩阵 $\boldsymbol{A}$ 的元素 a_{ij} 为

$$a_{ij}=\begin{cases}1, & s_i\boldsymbol{R}s_j \quad \boldsymbol{R}\text{ 表示 }s_i\text{ 与 }s_j\text{ 有关系}\\ 0, & s_i\bar{\boldsymbol{R}}s_j \quad \bar{\boldsymbol{R}}\text{ 表示 }s_i\text{ 与 }s_j\text{ 无关系}\end{cases} \tag{2.54}$$

例如图 2.15 中表示的一个有向图，可以用如下关联矩阵表示

$$\boldsymbol{A}=\begin{array}{c}\\ s_1\\ s_2\\ s_3\\ s_4\\ s_5\\ s_6\end{array}\begin{array}{c}\begin{array}{cccccc}s_1 & s_2 & s_3 & s_4 & s_5 & s_6\end{array}\\ \begin{bmatrix}0&0&0&0&0&0\\0&0&1&0&0&0\\1&1&0&0&0&0\\0&0&1&0&1&1\\1&0&0&0&0&0\\1&0&0&0&0&0\end{bmatrix}\end{array}$$

关联矩阵是布尔矩阵，这与 2.2.3 节中介绍的有一定区别，它们的运算遵守布尔代数的运算法则。有向图 $\boldsymbol{G}$ 与关联矩阵 $\boldsymbol{A}$ 之间有以下特性：

(1) 有向连接图 $\boldsymbol{G}$ 与关联矩阵 $\boldsymbol{A}$ 一一对应。

(2) 关联矩阵 $\boldsymbol{A}$ 中的元素全为 0 的列对应的节点为源点，或输入节点；元素全为 0 的

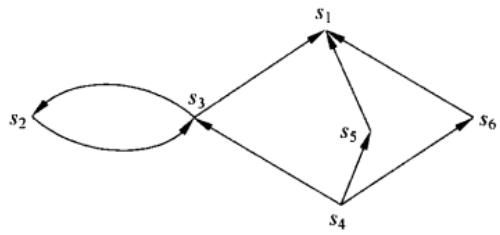


图 2.15 有向图与关联矩阵

行所对应的节点称为汇点，或输出节点。

(3) 在有向图 $\boldsymbol{G}$ 中，从节点 s_i 出发经过 k 条边可到达节点 s_j，则称节点 s_i 到节点 s_j 存在长度为 k 的通路，此时矩阵 $\boldsymbol{A}^k$ 的元素 $(i,j)=1$，否则为 0。

(4) 关联矩阵 $\boldsymbol{A}$ 表示其节点间是否存在有长度为 1 的通路，每个节点对应的行中，其元素为 1 的数量就是离开该节点的有向边数，每个节点对应的列中，其元素为 1 的数量就是进入该节点的有向边数。

(5) 由矩阵 $\boldsymbol{A}^k$ 的意义可知，矩阵 $\bigcup_{l=0}^{k}\boldsymbol{A}^l$ 表示了是否存在长度小于等于 k 的通路。

有向图 $\boldsymbol{G}=\{\boldsymbol{S},\boldsymbol{R}\}$ 中，对于 $s_i,s_j\in\boldsymbol{S}$，如果从节点 s_i 到节点 s_j 有任何一条通路存在，则称 s_i 可达 s_j。可达矩阵是指用矩阵形式来描述有向连接图各节点之间经过一定长度的通路后可达的程度，用 $\boldsymbol{R}$ 来表示，其定义为：有向图 $\boldsymbol{G}=\{\boldsymbol{S},\boldsymbol{R}\}$ 中的实体集合 $\boldsymbol{S}=\{s_1,s_2,\cdots,s_n\}$，则 $n\times n$ 矩阵 $\boldsymbol{R}$ 的元素 r_{ij} 为

$$r_{ij}=\begin{cases}1, & s_i \text{ 可达 } s_j\\ 0, & s_i \text{ 不可达 } s_j\end{cases},\quad \text{且 } r_{ii}=1(\text{即认为每个节点均自身可达}) \tag{2.55}$$

可达矩阵可根据关联矩阵计算出来

$$\boldsymbol{R}=\bigcup_{l=0}^{n-1}\boldsymbol{A}^l=(\boldsymbol{A}\cup\boldsymbol{I})^{n-1}=(\boldsymbol{A}\cup\boldsymbol{I})^n \tag{2.56}$$

利用可达矩阵可判断是否存在回路和构成回路的节点。在可达矩阵中，如果存在有不同节点对应的矩阵行和列都相同，则其有向图中的这些节点构成回路。

为计算可达矩阵，有时不用进行 n 次运算就可得到可达矩阵 $\boldsymbol{R}$，其计算步骤如下

$$\boldsymbol{A}_1=\boldsymbol{A}+\boldsymbol{I},\boldsymbol{A}_2=(\boldsymbol{A}+\boldsymbol{I})^2,\cdots,\boldsymbol{A}_{r-1}=(\boldsymbol{A}+\boldsymbol{I})^{r-1}$$

$$\boldsymbol{R}=\boldsymbol{A}_r=\boldsymbol{A}_{r-1}\neq\boldsymbol{A}_{r-2}\neq\cdots\neq\boldsymbol{A}_1,r\leqslant n-1$$

例 2.2 计算图 2.15 中有向图的可达矩阵

$$\boldsymbol{A}_1=\boldsymbol{A}+\boldsymbol{I}=\begin{array}{c}\\ s_1\\ s_2\\ s_3\\ s_4\\ s_5\\ s_6\end{array}\begin{bmatrix}1&0&0&0&0&0\\0&1&1&0&0&0\\1&1&1&0&0&0\\0&0&1&1&1&1\\1&0&0&0&1&0\\1&0&0&0&0&1\end{bmatrix}$$

$$\begin{array}{cccccc}s_1&s_2&s_3&s_4&s_5&s_6\end{array}$$

$$\boldsymbol{A}_2=(\boldsymbol{A}+\boldsymbol{I})^2=\begin{array}{c}\\ s_1\\ s_2\\ s_3\\ s_4\\ s_5\\ s_6\end{array}\begin{array}{c}\begin{array}{cccccc}s_1 & s_2 & s_3 & s_4 & s_5 & s_6\end{array}\\ \begin{bmatrix}1&0&0&0&0&0\\1&1&1&0&0&0\\1&1&1&0&0&0\\1&1&1&1&1&1\\1&0&0&0&1&0\\1&0&0&0&0&1\end{bmatrix}\end{array}$$

可以再计算 $\boldsymbol{A}_3=(\boldsymbol{A}+\boldsymbol{I})^2=\boldsymbol{A}_2$，所以可达矩阵 $\boldsymbol{R}=\boldsymbol{A}_2$。而且进一步观察，可以发现，节点 s_2 和节点 s_3 所对应的行和列均相同，则节点 s_2 和节点 s_3 构成一个回路。 ■

基于上述图论的基本原理，可建立便于计算机处理的各种结构模型。本节先从原理角度进行介绍，在 2.4 节及 2.5 节我们分别以电路系统模型设计和多体系统模型设计为例具体加以讨论。

一个系统中，实体 s_i 和 s_j 存在以下 4 种关系：

① $s_i \sim s_j$，s_i 与 s_j 和 s_j 与 s_i 互有关系，即关联矩阵元素 $a_{ij}=1, a_{ji}=1$。

② $s_i \times s_j$，s_i 与 s_j 和 s_j 与 s_i 均无关系，即关联矩阵元素 $a_{ij}=0, a_{ji}=0$。

③ $s_i > s_j$，s_i 与 s_j 有关系，s_j 与 s_i 无关系，即关联矩阵元素 $a_{ij}=1, a_{ji}=0$。

④ $s_i < s_j$，s_i 与 s_j 无关系，s_j 与 s_i 有关系，即关联矩阵元素 $a_{ij}=0, a_{ji}=1$。

至于关系的类型，不同系统有不同的约定，如因果关系、顺序关系、关联关系等，这些在进行模型设计时由用户加以确定，或者由支撑环境加以规定。

在进行模型设计时，要避免代数回路，可达矩阵为计算机检验代数回路提供了手段。经过检验，若存在回路，则必须去掉构成回路的某个元素。

在复杂系统结构化建模中，往往需要分层进行递阶模型设计。有向图可提供有效的计算机辅助方法，为此需要引入可达集和前因集概念。

对有向图中的每个节点 s_i，其可达节点汇集成一个集合，称为节点 s_i 的可达集 $\boldsymbol{R}(s_i)$，即可达矩阵中，节点 s_i 对应行所有矩阵元素为 1 的列所对应的节点集合。而将所有可能到达节点 s_i 的节点汇集成一个集合，称为节点 s_i 的前因集 $\boldsymbol{A}(s_i)$，即可达矩阵中，节点 s_i 对应列所有矩阵元素为 1 的行所对应的节点集合。用数学表达式表示为

$$\boldsymbol{R}(s_i)=\{s_j \in \boldsymbol{S} \mid r_{ij}=1\},\quad \boldsymbol{A}(s_i)=\{s_j \in \boldsymbol{S} \mid r_{ji}=1\} \tag{2.57}$$

其中 $\boldsymbol{S}$ 是关联矩阵的全部节点集合，r_{ij} 是可达矩阵的元素。

在进行多层模型设计时，假设不存在回路，其最高一层的节点不可能到达更高层的节点，它的可达集 $\boldsymbol{R}(s_i)$ 中只能是它本身自己，它的前因集 $\boldsymbol{A}(s_i)$ 则包括它自己和可到达的下级节点。这样，对于最高层节点，它的可达集 $\boldsymbol{R}(s_i)$ 与前因集 $\boldsymbol{A}(s_i)$ 的交集和它的可达集 $\boldsymbol{R}(s_i)$ 相同，如果不是最高层节点，它的可达集中则会有更高层的节点，这些节点不会出现在可达集与前因集的交集之内。因此，s_i 为最高层节点的充要条件是

$$\boldsymbol{R}(s_i)=\boldsymbol{R}(s_i) \cap \boldsymbol{A}(s_i) \tag{2.58}$$

得到最高层节点后，去掉可达矩阵中这些节点的对应行和列，按上述方法可继续寻找次高层的节点。依此类推，就可找到关联矩阵的各级节点。设 $\boldsymbol{L}_1, \boldsymbol{L}_2, \cdots, \boldsymbol{L}_k$ 表示层次结构中从上到下的各级，级别的划分可用下式表示

$$\prod(\boldsymbol{S}) = [\boldsymbol{L}_1, \boldsymbol{L}_2, \cdots, \boldsymbol{L}_k] \tag{2.59}$$

上述方法可用迭代算法来实现，为此定义第 0 级 $\boldsymbol{L}_0 = \boldsymbol{\Phi}$，是个空集，则迭代算法可表示如下

$$\boldsymbol{L}_k = \{s_i \in \boldsymbol{S} - \boldsymbol{L}_0 - \boldsymbol{L}_1 - \cdots - \boldsymbol{L}_{k-1} \mid \boldsymbol{R}_{k-1}(s_i) \cap \boldsymbol{A}_{k-1}(s_i)\} \tag{2.60}$$

其中，$\boldsymbol{R}_{k-1}(s_i)$ 与 $\boldsymbol{A}_{k-1}(s_i)$ 分别表示从 $(\boldsymbol{S} - \boldsymbol{L}_0 - \boldsymbol{L}_1 - \cdots - \boldsymbol{L}_{k-1})$ 子集中求得的 s_i 的可达集和前因集。

有了可达树和层次的划分，就可以进行结构模型的设计了，其步骤如下：

① 根据层次划分结果，重新排列去除回路后的可达矩阵，即可达矩阵的行与列对应节点均按层次级别的顺序 $\boldsymbol{L}_1, \boldsymbol{L}_2, \cdots, \boldsymbol{L}_k$ 顺序排列，从而形成新的可达矩阵。

② 按 $\boldsymbol{L}_1, \boldsymbol{L}_2, \cdots, \boldsymbol{L}_k$ 顺序得到每一级别中的节点，位于有向连接图同一水平。

③ 按新的可达矩阵，找出两级关系分块矩阵中“1”元素对应的节点对，从上到下画出相邻两级的带箭头的连线。

④ 跨级连线的方法同③，但需首先判断这条线是否可以由传递性推出，若能，则不需画这条连线。

⑤ 恢复有向图中因为构成回路而去掉的节点，同时恢复原来的连接关系。

2.4 电路系统模型设计

2.3 节讨论了图论用于模型设计的基本思想，本节以电路系统为例，进一步讨论图论用于计算机辅助模型设计的技术和方法。

电路仿真计算与分析是十分常见的，典型的任务包括电路稳态计算、灵敏度分析、频率分析和时域分析等。电路稳态计算包括直流解（网络的工作点）和交流稳态解（频率响应）。灵敏度分析不但要考虑电路元件参数变化对系统性能的影响，还要考虑寄生参数对系统性能的影响。电路的频率分析就是要考察电路网络动态特性，其目标是将电路网络进行线性化近似后，计算网络的传递函数，包括其零极点的分布及其灵敏度。时域分析的目标是要考察电路网络在某种输入或初态下网络随时间变化的性能。

显然，如何方便建立电路模型，并在此基础上进行求解，是电路分析与设计的基本要求。传统上大多采用先搭建实验板进行物理模型设计与分析，这本质上是一种物理仿真。随着大规模集成电路的发展，基于计算机仿真的电路分析与设计变得日益迫切，因为任何传统的方法已经无能为力了。

2.4.1 电路系统模型

电路模型首先要基于电路基本定律描述，然后要考虑在计算机上实现的方便性。基于基尔霍夫电流定律（KCL）和基尔霍夫电压定律（KVL），元件的支路方程易于得到，下面以由 R、L、C 元件和独立源（电流源或电压源）组成的线性时不变电路加以说明。

基尔霍夫电流定律是：从任一节点流出的电流的代数和等于零。根据该定律，对一个有 $n+1$ 个端点、一个参考节点（编号为 0）的电路，由 R、L、C 元件连接。假设各端点的电压

用 $V_i(i=1,2,\cdots,n)$ 来表示，其值是相对于地测量的。电压源或电流源可接到任意一个端点与参考节点之间(参见图 2.16)。

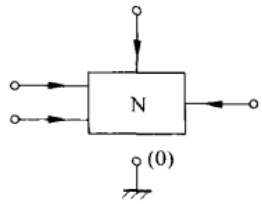


图 2.16 具有 $n+1$ 个节点的电路

根据戴维南及诺顿原理，电压源与电流源是可以转换的，因此，由 R、L、C 元件及电流源连接构成的电路具有一般性。显然，在任意节点，流过无源元件的电流等于由独立电流源产生的电流，即

$$\sum(\text{通过无源元件流出节点的电流}) = \sum(\text{从独立电源流入节点的电流})$$

则电路的特性可用如下矩阵方程来描述

$$\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1,n+1} \\ y_{21} & y_{22} & \cdots & y_{2,n+1} \\ \vdots & \vdots & & \vdots \\ y_{n+1,1} & y_{n+1,1} & \cdots & y_{n+1,n+1} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_{n+1} \end{bmatrix} = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_{n+1} \end{bmatrix}$$

简记为

$$\boldsymbol{YV} = \boldsymbol{J} \tag{2.61}$$

其中 $J_i(i=1,2,\cdots,n+1)$ 是从参考节点流入第 i 个节点的电流，通常表示电流源，y_{ij} 为节点 j 与 i 之间的导纳，这种方法称为节点法。

$\boldsymbol{Y}$ 阵与 $\boldsymbol{J}$ 向量具有以下特性：

(1) $\boldsymbol{Y}$ 的各对角元素为正，并且：$y_{jj} = \sum$ 接在节点 j 的导纳。

(2) $\boldsymbol{Y}$ 的各非对角元素是负的，并且 $y_{ij} = -\sum$ 接在节点 i 和 j 之间的导纳。

(3) 右端向量 $\boldsymbol{J}$ 的第 j 个分量为 $J_j = \sum$ 从独立电源流向节点 j 的电流。

(4) 电路中第 i 个元件如果位于节点 j 和节点 k 之间，则其导纳为

$$\boldsymbol{Y} = \sum_i y_i (e_{j_i} - e_{k_i})(e_{j_i} - e_{k_i})^{\mathrm{T}}$$

其中 y_i 表示接在节点 j_i 和 k_i 之间的元件 i 的导纳，e_j 是第 j 个分量为 1 而其他分量为 0 的单位向量，也可认为是单位矩阵的第 j 列。

(5) 当电路中有多个电流源时，$\boldsymbol{J}$ 向量可以表示为

$$\boldsymbol{J} = -\sum_i J_i (e_{j_i} - e_{k_i})$$

这样就能以元件的编号的方式得出 $\boldsymbol{Y}$ 矩阵。

例 2.3 考虑如图 2.17 所示的电路，用节点电压和元件的导纳表示无源元件的电流，可得到

节点 1 $G_1(V_1-V_4)+(sC_4+G_7)(V_1-V_2)+G_6(V_1-V_3)=J_1-J_6+J_4$

节点 2 $(sC_4+G_7)(V_2-V_1)+\dfrac{1}{sL_2}(V_2-V_4)+sC_5(V_2-V_3)=-J_4$

节点 3 $\dfrac{1}{sL_3}(V_3-V_4)+sC_5(V_3-V_2)+G_6(V_3-V_1)=J_3+J_6$

节点 4 $G_1(V_4-V_1)+\dfrac{1}{sL_2}(V_4-V_2)+\dfrac{1}{sL_3}(V_4-V_3)=-J_1-J_3$

重新整理，得到节点方程的形式为

$$
\begin{array}{c}
\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix}
\begin{bmatrix}
G_1+sC_4+G_6+G_7 & -sC_4-G_7 & -G_6 & -G_1 \\
-sC_4-G_7 & \frac{1}{sL_2}+sC_4+sC_5+G_7 & -sC_5 & -\frac{1}{sL_2} \\
-G_6 & -sC_5 & \frac{1}{sL_3}+sC_5+G_6 & -\frac{1}{sL_3} \\
-G_1 & -\frac{1}{sL_2} & -\frac{1}{sL_3} & G_1+\frac{1}{sL_2}+\frac{1}{sL_3}
\end{bmatrix}
\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} \\
=\begin{bmatrix} J_1+J_4-J_6 \\ -J_4 \\ J_3+J_6 \\ -J_1-J_3 \end{bmatrix}
\end{array}
$$

观察该节点方程左边的系数矩阵 $\boldsymbol{Y}$ 和右边的电流源 $\boldsymbol{J}$ 向量的元素，显然满足前面所述的特点。 ■

2.4.2 电路系统的拓扑

对于图 2.17 所示的电路，当然我们可以通过人工的办法来建立节点方程。然而系统规模增大，需要解决如何利用电路的拓扑关系来自动建立模型，即确定 $\boldsymbol{Y}$ 矩阵。

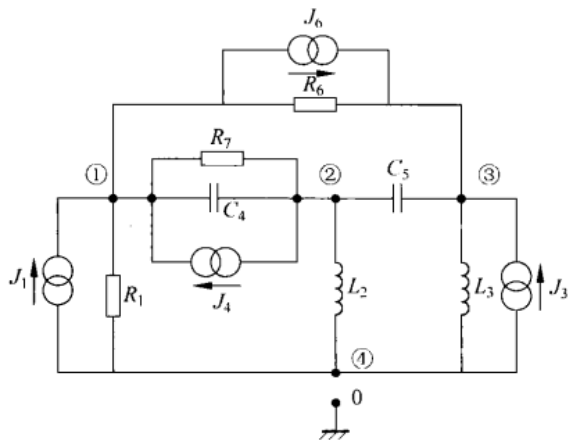


图 2.17 具有 4 个电流源的网络

将图论用于不同领域的模型设计时，模型的表示方法不尽相同。在电路系统模型设计时，其拓扑图一般用“边”表示两端元件，代表电路系统内部的连接或拓扑关系。电路中的节点标以号码放在圆圈中。网络方程基于基尔霍夫定律定义，既可以是基于电压定律，也可以是基于电流定律。每条边上标以方向，代表电流的方向，或者说是电压下降的方向。当然这只是一个设定方向，实际方向可能正好相反，这由求得的值来决定，如果为正值，则设定的方向与实际一致，如果为负，则设定的方向与实际相反。

用有向图表示一个电路系统是很方便的。例如图 2.18 所示的电路系统，其有向图可直接给出，如图 2.19 所示。

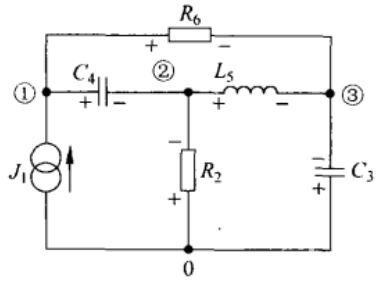


图 2.18 一个简单的电路系统图

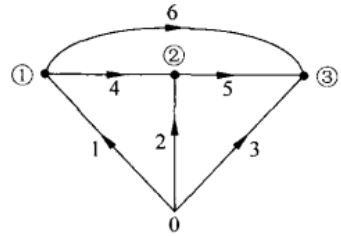


图 2.19 图 2.18 电路的有向图

该电路的关联矩阵的定义与 2.3.1 节有一些区别，它考虑节点间必须遵循的物理定律，如基尔霍夫电流定律(KCL)，按照电路图规定的方向(从节点流出的电流定义为正方向)，可列出 3 个节点的电流方程(0 是对地节点，是参考节点)

$$\text{节点 1：} -i_1 + i_4 + i_6 = 0$$
$$\text{节点 2：} -i_2 - i_4 + i_5 = 0$$
$$\text{节点 3：} -i_3 - i_5 - i_6 = 0$$

写成矩阵形式

$$\boldsymbol{AI} = \boldsymbol{0}$$

其中 $\boldsymbol{A}$ 称为关联矩阵

$$\boldsymbol{A} = \begin{array}{c} \\ \text{节点}\downarrow \end{array} \begin{array}{c} \text{边}\rightarrow \\ \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 1 \\ 0 & -1 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

考虑电流与电压的关系，若用 v_{ni} 表示第 i 个节点相对于参考节点的电压，我们不难得到每条边的电压 v_j 表达式如下

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} v_{n1} \\ v_{n2} \\ v_{n3} \end{bmatrix}$$

不难看出

$$\boldsymbol{V} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{V}_{\mathrm{n}} \tag{2.62}$$

假设电路系统中所有的电压源均转换为电流源。进一步，在节点法中，将无源元件的电流用 $\boldsymbol{I}_{\mathrm{b}}$ 表示，而电流源的电流用 $\boldsymbol{J}_{\mathrm{b}}$ 表示，则节点电流方程可表示为

$$[\boldsymbol{A} \quad \boldsymbol{A}_{\mathrm{J}}] \begin{bmatrix} \boldsymbol{I}_{\mathrm{b}} \\ \boldsymbol{J}_{\mathrm{b}} \end{bmatrix} = 0$$

即

$$\boldsymbol{A}\boldsymbol{I}_{\mathrm{b}} = -\boldsymbol{A}_{\mathrm{J}}\boldsymbol{J}_{\mathrm{b}} \tag{2.63}$$

从电路系统模型可知，对每个无源元件的电流若以 $Y_{bi}V_{bi} = I_{bi}$ 的形式描述，则全部无源

元件的电流可表示为 $\boldsymbol{Y}_b\boldsymbol{V}_b=\boldsymbol{I}_b$，代入式(2.63)可得

$$\boldsymbol{A}\boldsymbol{Y}_b\boldsymbol{V}_b=-\boldsymbol{A}_J\boldsymbol{J}_b$$

注意，此时的 $\boldsymbol{Y}_b$ 是无源支路的导纳矩阵。进一步讨论 KVL 的表达形式，将无源元件的电压用 $\boldsymbol{V}_b$ 表示，而电流源两端的电压用 $\boldsymbol{V}_J$ 表示，则节点电压方程可表示为

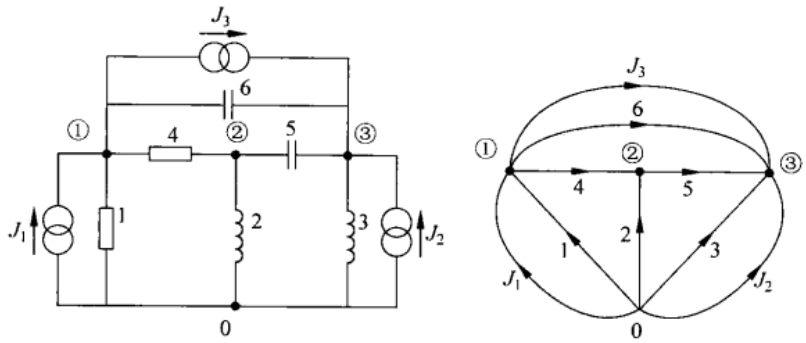
$$\begin{bmatrix}\boldsymbol{V}_b\\ \boldsymbol{V}_J\end{bmatrix}=\begin{bmatrix}\boldsymbol{A}^T\\ \boldsymbol{A}_J^T\end{bmatrix}\boldsymbol{V}_n \tag{2.64}$$

分别得到 $\boldsymbol{V}_b=\boldsymbol{A}^T\boldsymbol{V}_n$，$\boldsymbol{V}_J=\boldsymbol{A}_J^T\boldsymbol{V}_n$，将 $\boldsymbol{V}_b$ 代入 KCL 的节点方程可得

$$\boldsymbol{A}\boldsymbol{Y}_b\boldsymbol{A}^T\boldsymbol{V}_n=-\boldsymbol{A}_J\boldsymbol{J}_b \tag{2.65}$$

注意 $\boldsymbol{A}\boldsymbol{Y}_b\boldsymbol{A}^T$ 就是在电路系统模型 2.4.1 节讨论过的节点导纳矩阵 $\boldsymbol{Y}$。这样，只要输入关联矩阵 $\boldsymbol{A}$ 和无源支路的导纳矩阵 $\boldsymbol{Y}_b$，就可以计算出节点导纳矩阵 $\boldsymbol{Y}$。

例 2.4 如图 2.20(a)所示的电路系统，为建立其模型，先确定其关联矩阵，可通过有向图来得到(参见图 2.20(b))。



(a) 3节点电路系统　　(b) 3节点电路系统有向图

图 2.20

从有向图很容易得到

$$\boldsymbol{Y}_b=\begin{bmatrix}G_1 & & & & & \\ & \frac{1}{s}L_2 & & & & \\ & & \frac{1}{s}L_3 & & & \\ & & & G_4 & & \\ & & & & sC_5 & \\ & & & & & sC_6\end{bmatrix}$$

$$\boldsymbol{A}=\begin{array}{c}\begin{matrix}1 & 2 & 3 & 4 & 5 & 6\end{matrix}\\ \begin{bmatrix}1 & 0 & 0 & 1 & 0 & 1\\ 0 & 1 & 0 & -1 & 1 & 0\\ 0 & 0 & 1 & 0 & -1 & -1\end{bmatrix}\end{array},\quad \boldsymbol{A}_J=\begin{array}{c}\begin{matrix}J_1 & J_2 & J_3\end{matrix}\\ \begin{bmatrix}-1 & 0 & 1\\ 0 & 0 & 0\\ 0 & -1 & -1\end{bmatrix}\end{array}$$

从而可由矩阵计算得到

$$\boldsymbol{Y}=\boldsymbol{A}\boldsymbol{Y}_b\boldsymbol{A}^T=\begin{bmatrix}G_1+G_4+sC_6 & -G_4 & -sC_6\\ -G_4 & G_4+sC_5+\frac{1}{s}L_2 & -sC_5\\ -sC_6 & -sC_5 & sC_5+sC_6+\frac{1}{s}L_3\end{bmatrix}$$

令 $\boldsymbol{J}_n=-\boldsymbol{A}_1\boldsymbol{J}_b$，则

$$\boldsymbol{J}_n=\begin{bmatrix}1&0&-1\\0&0&0\\0&1&1\end{bmatrix}\begin{bmatrix}J_1\\J_2\\J_3\end{bmatrix}=\begin{bmatrix}J_1-J_3\\0\\J_2+J_3\end{bmatrix}$$

■

2.4.3 改进节点法

在前面讨论的节点法中，要求电路元件具有导纳形式。但是有一些元件，如电压源、电压控制的电压源、电流控制的电流源、变压器、理想变压器、运算放大器、任意值器、零值器等不存在导纳形式，为此需要对这种节点法加以改进。

设电路有 b 个支路，$n+1$ 个节点，包含 R,G,L,C 元件及电源。利用关联矩阵，KCL 及 KVL 方程组为

$$\boldsymbol{A}\boldsymbol{I}_b=0,\quad \boldsymbol{V}_b-\boldsymbol{A}^{\mathrm{T}}\boldsymbol{V}_n=0$$

下标 n 表示节点，下标 b 表示支路，在矩阵中就是元件号。

这里引入表示支路方程的通用形式

$$\begin{matrix}\text{电流}\\\text{电压}\end{matrix}\begin{bmatrix}\boldsymbol{Y}_1\\K_2\end{bmatrix}\boldsymbol{V}_b+\begin{bmatrix}K_1\\\boldsymbol{Z}_2\end{bmatrix}\boldsymbol{I}_b=\begin{bmatrix}W_{b_1}\\W_{b_2}\end{bmatrix}\tag{2.66}$$

其中 $\boldsymbol{Y}_1,\boldsymbol{Z}_2$ 分别代表导纳和阻抗，K_1,K_2 为无量纲常数，W_{b_1},W_{b_2} 包括独立电流源与电压源以及初始条件对电容和电感的影响。为了简单起见，用下式表示电流与电压关系

$$\boldsymbol{Y}_b\boldsymbol{V}_b+\boldsymbol{Z}_b\boldsymbol{I}_b=\boldsymbol{W}_b\tag{2.67}$$

其中电容以导纳形式表示，电感以阻抗形式表示，这样得到一组微分-代数方程。表 2.1 列出了典型两端元件的支路方程及 $\boldsymbol{Y}_b,\boldsymbol{Z}_b,\boldsymbol{W}_b$。

表 2.1 典型元件支路的方程及参数

元件	支路方程	Y_b	Z_b	W_b
电阻	$V_b-R_bI_b=0$	1	$-R_b$	0
电导	$G_bV_b-I_b=0$	G_b	-1	0
电容	$sC_bV_b-I_b=C_bV_0$	sC_b	-1	C_bV_0
电感	$V_b-sL_bI_b=-L_bI_0$	1	$-sL_b$	$-L_bI_0$
电压源	$V_b=E_b$	1	0	E_b
电流源	$I_b=J_b$	0	1	J_b

改进节点法的基本思路是将电路元件进行分组，具有导纳形式的元件分为一组，仍采用节点法。对 KCL 方程，对电路元件进行排序，可以用下式表示

$$\begin{bmatrix}\boldsymbol{A}_1&\boldsymbol{A}_2&\boldsymbol{A}_3\end{bmatrix}\begin{bmatrix}\boldsymbol{I}_1\\\boldsymbol{I}_2\\\boldsymbol{J}\end{bmatrix}=0$$

其中 $\boldsymbol{I}_1$ 包括具有导纳形式的元件的支路电流以及不要求求解的支路电流，$\boldsymbol{I}_2$ 包括没有导纳形式的元件的支路电流以及电压源的支路电流，还包括要求求解的支路电流，$\boldsymbol{J}$ 为独立电

流源。

KVL 方程可按同样的办法划分，其表达式如下

$$\begin{bmatrix} \boldsymbol{V}_1 \\ \boldsymbol{V}_2 \\ \boldsymbol{V}_J \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_1^{\mathrm{T}} \\ \boldsymbol{A}_2^{\mathrm{T}} \\ \boldsymbol{A}_3^{\mathrm{T}} \end{bmatrix} \boldsymbol{V}_n$$

显然，一旦 $\boldsymbol{V}_n$ 求出，就可计算出 $\boldsymbol{V}_J$。

第一部分的支路关系为 $\boldsymbol{Y}_1\boldsymbol{V}_1=\boldsymbol{I}_1$，将其代入改进节点法的 KCL 方程，不难得到

$$\boldsymbol{A}_1\boldsymbol{Y}_1\boldsymbol{V}_1 + \boldsymbol{A}_2\boldsymbol{I}_2 = -\boldsymbol{A}_3\boldsymbol{J}$$

再将改进节点法的 KVL 方程的 $\boldsymbol{V}_1$ 表达式代入，可得

$$\boldsymbol{A}_1\boldsymbol{Y}_1\boldsymbol{A}_1^{\mathrm{T}}\boldsymbol{V}_n + \boldsymbol{A}_2\boldsymbol{I}_2 = -\boldsymbol{A}_3\boldsymbol{J}$$

令 $\boldsymbol{A}_1\boldsymbol{Y}_1\boldsymbol{A}_1^{\mathrm{T}}=\boldsymbol{Y}_m$，$-\boldsymbol{A}_3\boldsymbol{J}=\boldsymbol{J}_n$，则上式记为

$$\boldsymbol{Y}_m\boldsymbol{V}_n + \boldsymbol{A}_2\boldsymbol{I}_2 = \boldsymbol{J}_n$$

显然，$\boldsymbol{Y}_m$ 是第一部分中元件的节点导纳，$\boldsymbol{J}_n$ 代表等效节点电流。

改进节点法的 KVL 方程第二部分中，其支路方程具有如下形式

$$\boldsymbol{Y}_2\boldsymbol{V}_2 + \boldsymbol{Z}_2\boldsymbol{I}_2 = \boldsymbol{W}_2$$

其中 $\boldsymbol{W}_2$ 只包括对应于电压源的非零分量。这样将改进节点法的 KVL 方程第二部分的 $\boldsymbol{V}_2$ 代入，可得

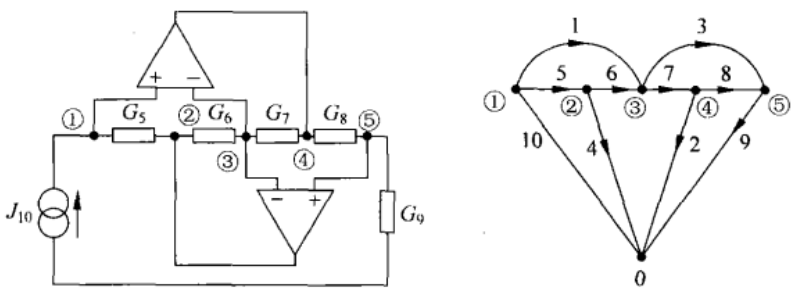
$$\boldsymbol{Y}_2\boldsymbol{A}_2^{\mathrm{T}}\boldsymbol{V}_n + \boldsymbol{Z}_2\boldsymbol{I}_2 = \boldsymbol{W}_2$$

将第一部分与第二部分的方程写在一起，从而得到改进节点法的模型描述形式

$$\begin{array}{ll} \text{KCL} & \\ \text{附加方程} & \end{array} \begin{bmatrix} \boldsymbol{Y}_m & \boldsymbol{A}_2 \\ \boldsymbol{Y}_2\boldsymbol{A}_2^{\mathrm{T}} & \boldsymbol{Z}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_n \\ \boldsymbol{I}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_n \\ \boldsymbol{W}_2 \end{bmatrix} \begin{array}{l} \text{接到节点上的电流源} \\ \text{电压源的影响} \end{array} \tag{2.68}$$

这样就得到改进节点法的电路模型。

例 2.5 运算放大器可用来做通用阻抗变换器如图 2.21(a)所示，试用改进节点法给出其模型的方程。



(a) 通用阻抗变换器　　(b) 通用阻抗变换器有向图

图 2.21　通用阻抗变换器及其有向图

运算放大器的符号如图 2.22 所示，其支路方程如下

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

由此不难得到其有向图如图 2.21(b)所示。

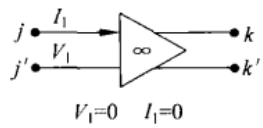


图 2.22　理想运算放大器支路方程

两个运算放大器用前 4 个支路表示，支路方程为

$$\begin{bmatrix}1&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&0\end{bmatrix}\begin{bmatrix}V_1\\V_2\\V_3\\V_4\end{bmatrix}+\begin{bmatrix}0&0&0&0\\1&0&0&0\\0&0&0&0\\0&0&1&0\end{bmatrix}\begin{bmatrix}I_1\\I_2\\I_3\\I_4\end{bmatrix}=\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$$

子矩阵 $\boldsymbol{A}_2$ 为如下形式

$$\boldsymbol{A}_2=\begin{array}{c}\text{边}\rightarrow\\ \text{节点}\downarrow\end{array}\begin{array}{c}\begin{array}{cccc}1&2&3&4\end{array}\\ \begin{bmatrix}1&0&0&0\\0&0&0&1\\-1&0&1&0\\0&1&0&0\\0&0&-1&0\end{bmatrix}\end{array}\begin{array}{c}1\\2\\3\\4\\5\end{array}$$

从而得到改进节点法给出的模型为如下形式

$$\left[\begin{array}{ccccc:cccc}G_5&-G_5&0&0&0&1&0&0&0\\-G_5&G_5+G_6&-G_6&0&0&0&0&0&1\\0&-G_6&G_6+G_7&-G_7&0&-1&0&1&0\\0&0&-G_7&G_7+G_8&-G_8&0&1&0&0\\0&0&0&-G_8&G_8+G_9&0&0&-1&0\\1&0&-1&0&0&0&0&0&0\\\hdashline 0&0&0&0&0&1&0&0&0\\0&0&1&0&-1&0&0&0&0\\0&0&0&0&0&0&0&1&0\end{array}\right]\left[\begin{array}{c}V_{n_1}\\V_{n_2}\\V_{n_3}\\V_{n_4}\\V_{n_5}\\\hdashline I_1\\I_2\\I_3\\I_4\end{array}\right]=\left[\begin{array}{c}J_{10}\\0\\0\\0\\0\\\hdashline 0\\0\\0\\0\end{array}\right]$$

■

2.4.4 小结

电路系统仿真归根结底是要得到网络的微分-代数方程组。一般说来，单个元件的模型是可以得到的(无论是理论模型还是实验模型)，但将其构成大规模电路网络后，就必须借助于计算机辅助建模手段。基于有向图和关联矩阵的计算机辅助建模为电路系统模型设计提供了有效工具。现代的商用电路仿真软件大都提供了各种电路元件的模型库，以及基于图论的模型设计工具，使得用户只需选择相应的元件、正确输入它们之间的连接关系，计算机会自动生成微分-代数方程组，为电路计算、容差分析、频率分析以及时域分析提供基本模型。当然，本节只介绍了基于图论建模的基本理论与方法，将其用于实际问题时还需要进一步完善，包括提高处理的速度、减少内存的占用以及根据仿真的目标在基本模型基础上做进一步的设计等。限于篇幅，本书不作进一步讨论，有兴趣的读者可参考其它相关书籍。

2.5 多体系统模型设计*

工程领域中的机械系统是由大量零部件构成的，在对这些复杂系统进行设计优化与性能分析时，常分为两大类，一类称为结构，它们的特征是在正常的工况下构件间没有相对运动，如房屋建筑、桥梁、航空航天器与各种车辆的壳体以及各种零部件的本身，人们关心的是这些结构在受到载荷时的强度、刚度与稳定性。

另一类称为机构，是装配在一起并允许作相对运动的若干个刚体的组合，其特征是系统在运动过程中这些部件间存在相对运动（参见图 2.23）。

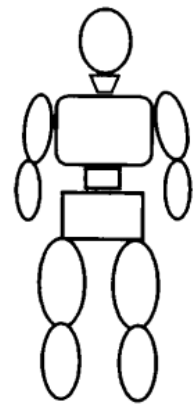


图 2.23 机器人的多体系统模型

上述复杂系统的力学模型为多个物体通过运动副连接的系统，称为多体系统。对于复杂多体机械系统，问题大致有三类。

一是在不考虑系统运动起因的情况下研究各部件的位置与姿态以及它们的变化速度与加速度的关系，称为系统的运动学分析；

二是当系统受到静载荷时，确定在运动副制约下的系统平衡位置以及运动副静反力，这类问题称为系统的静力学分析；

三是讨论载荷与系统运动的关系，研究复杂系统在载荷作用下各部件的动力学响应即动力学问题。

已知系统的运动确定运动副的动反力的问题是系统各部件强度分析的基础，这类问题称为动力学的逆问题。现代机械系统离不开控制技术，即系统的部分构件受控，当它们按已知规律运动时，研究外载荷作用下系统其他构件如何运动，这类问题称为动力学正逆混合问题。

多体系统动力学其主要任务是：建立复杂机械系统运动学和动力学形式化的数学模型，开发实现这个数学模型的软件系统，用户只需输入描述系统的最基本数据，借助计算机就能自动进行程式化处理；开发和实现有效的处理数学模型的计算方法与数值积分方法，自动得到运动学规律和动力学响应；实现有效的数据后处理，采用动画显示、图表或其他方式提供数据处理结果。

2.5.1 多体系统建模基本概念

1. 物体

多体系统中的构件定义为物体，分为刚体与柔体。刚体是指质点间距离保持不变的质点系，而柔体指考虑质点间距离变化的质点系。运动学分析中，通常将那些运动性能需要特别关注的零部件定义为物体。在动力学分析中，物体的惯量特性是影响系统的重要参数，因此部件惯量的大小成了是否将部件定义成物体的一个重要依据。一个低速运动的对象，其零部件的弹性变形并不影响其大范围的运动性态，可以做刚性假定；高速情况下，大范围运

动与弹性变形耦合容易引起复杂的动力学性态，物体必须做柔性假定。刚柔混合多体系统是多体系统中最常见的模型。

2. 铰

多体系统中物体间的运动约束定义为铰，也称为运动副。机构本身的运动副是铰的物理背景，另外，还包括一些非机构运动副的运动约束，如曲柄滑块机构中的滑块等。铰连接的一对物体称为该铰的邻接物体。忽略铰的质量，将标志铰的几何点称为铰点。

3. 外力(外力偶)

多体系统外的物体对系统中物体的作用定义为外力(外力偶)。重力是典型的外载荷。对刚性体，力偶可以作用在构件的任意点上；对柔性体，由于要考虑对构件弹性变形的影响，力偶必须施加在合适的作用点上。

4. 力元

多体系统中物体之间的相互作用定义为力元，也称内力。实际系统中零部件之间的相互联系，一种是通过运动副，另一种通过力的相互作用。两者的本质差异为，前者限制了相连物体相对运动的自由度。后者却没有这种限制，力元的作用必须通过物体来实现，它描述了构件之间的相互作用方式和作用力大小。

5. 坐标系

直角坐标系：基本坐标系，以$\{0,x,y,z\}$表示。

拉氏坐标系：对直角坐标系变换所得的坐标系。

全局坐标系：也称绝对坐标系或固定坐标系，有作为参照的固定原点和方向。

连体坐标系：固定在刚体上并随其运动的坐标系，用以确定刚体运动。

广义坐标系：唯一地确定机构所有构件位置和方位即机构构形的任意一组变量的坐标系。

2.5.2 多体系统拓扑构型

多体系统内各物体的联系方式称为系统的拓扑构型，简称拓扑。利用图论的几何方法，可对多体系统的拓扑构型做出直观的描述。每个物体记作$B_i(i=1,2,\cdots,n)$，i表示物体的序号，n为系统中物体的个数。铰用一条连接邻接物体的有向线段表示，记作$H_j(j=1,2,\cdots)$，j表示铰的序号，也同时描述了铰点所在的位置。

用有向线段表示铰，其目的有两个：一个是用以明确刚体之间相互作用力的正方向，一个是定义相邻物体相对运动的参照关系。这种由顶点和线段组成的有向图称为多体系统的结构图。

铰与邻接物体的关系称为关联。设顶点B_i沿一系列物体和铰到达另一顶点B_j而没有一个铰被重复过，则称这组铰(或物体)组成B_i至B_j的路。当系统中任意两顶点之间只有唯一的路存在时，称为树系统，反之为非树系统或带回路系统。图2.24所示便是一树系统

的拓扑结构图。如果在 B_1 和 B_5 之间多增加一个铰 H_6，则变成了图 2.25 所示的带回路系统。

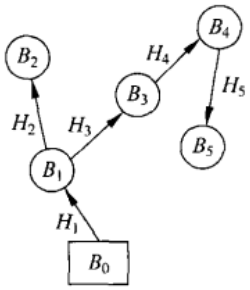


图 2.24 一个树系统的拓扑结构图

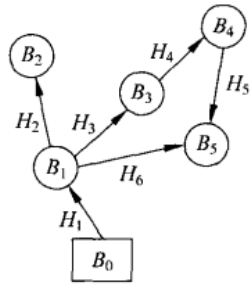


图 2.25 一个带回路系统的拓扑结构图

工程实际中，常见的机器人手臂（见图 2.26）便是一个树系统的典型例子。非树系统的一个常见例子是一个曲柄滑块机构，如图 2.27 所示。图中除了根物体以外有两个物体，三个铰，铰的个数多于物体的个数，为非树系统。

人为解除非树系统中某些铰的约束，可使原来的非树系统转变为树系统，称为原系统的派生树系统。

如果多体系统力学模型与系统外运动规律已知的物体有铰联系，则称该系统为有根系统。B_0 表示系统外运动为已知的物体，称为系统的根。具有固定基座的任何机械系统都是有根系统，图 2.26 和图 2.27 中的系统都是有根系统。

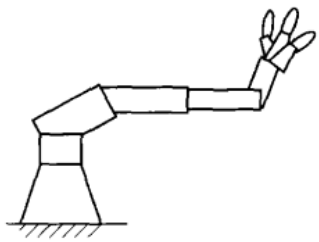


图 2.26 典型的树系统——机器人手臂

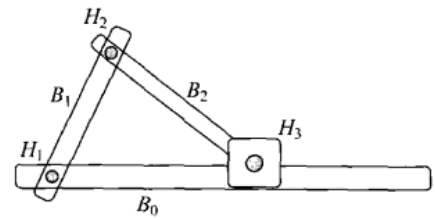


图 2.27 一个简单的非树系统——曲柄滑块机构

如果多体系统力学模型与系统外运动规律为已知的物体无任何铰联系，则称该系统为无根系统。无根系统的一个典型例子是卫星天线系统。无根系统可以将参考坐标视作抽象的根，并认为系统以抽象的虚铰与其根相联系，则有根系统和无根系统可以在形式上取得一致。

设有 n 个物体 $B_i(i=1,2,\cdots,n)$ 和 n 个铰 $H_j(j=1,2,\cdots,n)$ 组成的树系统，规定：若 B_j 在 B_0 至 B_i 的路上，则称 B_j 位于 B_i 的内侧，或 B_i 位于 B_j 的外侧。与任意物体 B_i 直接联系的内侧物体称为 B_i 的内接物体，联系铰称为 B_i 的内接铰。物体和铰按以下规则标号：

① 各物体的编号大于其内接物体的序号；

② 各物体与其内接铰有相同的序号；

③ 表示铰的有向线段由内侧指向外侧。

按此规定，任意铰 H_j 的外接物体的序号等于铰的序号 j，即 H_j 的外接物体为 B_j。所有内接物体的序号 i 定义为 j 的整标函数 $i(j)$，且有 $i(j)<j$，i 为 H_j 的内接物体的序号。

$i(j)(j=1,2,\cdots,n)$称为该系统的内接数组，是对系统结构的一种数学描述。例如，图 2.24 树系统的内接数组可以列写为表 2.2。

表 2.2 图 2.24 内接数组

j	1	2	3	4	5
$i(j)$	0	1	1	3	4

引入两个一维整型数组：$i^+(j)$和$i^-(j)$ $(j=1,\cdots,n)$。$i^+(j)$和i (j)的值为与铰 H_j 关联的两个物体的标号，且该铰由物体 $B_{i^+(j)}$ 指向物体 $B_{i^-(j)}$，则 $i^+(j)$和$i^-(j)$称为关联数组，它与系统的拓扑结构图一一对应(参见表 2.3)。

表 2.3 图 2.24 的关联数组

j	1	2	3	4	5
$i^+(j)$	0	1	1	3	4
$i^-(j)$	1	2	3	4	5

关联矩阵和通路矩阵是对系统结构的另一种数学描述。关联矩阵的行号和列号分别与物体和铰的序号相对应，其第 i 行第 j 列元素 S_{ij} 定义为(参见表 2.4)

表 2.4 图 2.24 关联矩阵

i \ j	1	2	3	4	5
0	1	0	0	0	0
1	−1	1	1	0	0
2	0	−1	0	0	0
3	0	0	−1	1	0
4	0	0	0	−1	1
5	0	0	0	0	−1

$$S_{ij}=\begin{cases}1, & \text{当 } H_j \text{ 与 } B_i \text{ 关联且以 } B_i \text{ 为起点}\\ -1, & \text{当 } H_j \text{ 与 } B_i \text{ 关联且以 } B_i \text{ 为终点}\\ 0, & \text{当 } H_j \text{ 与 } B_i \text{ 无关联}\end{cases}$$

通路矩阵 $\boldsymbol{T}$ 为 n 阶方阵，与关联矩阵相反，其行号和列号分别与铰和物体的序号相对应，其第 j 行第 i 列元素 T_{ji} 定义为(参见表 2.5)

表 2.5 图 2.24 通路矩阵

j \ i	1	2	3	4	5
1	−1	−1	−1	−1	−1
2	0	−1	0	0	0
3	0	0	−1	−1	−1
4	0	0	0	−1	−1
5	0	0	0	0	−1

$$T_{ji}=\begin{cases}1, & \text{当 } H_j \text{ 属于 } B_0 \text{ 至 } B_i \text{ 的路且指向 } B_0\\ -1, & \text{当 } H_j \text{ 属于 } B_0 \text{ 至 } B_i \text{ 的路且背向 } B_0\\ 0, & \text{当 } H_j \text{ 不属于 } B_0 \text{ 至 } B_i \text{ 的路}\end{cases}$$

$\boldsymbol{S}$ 和 $\boldsymbol{T}$ 矩阵有以下的性质：

① $\boldsymbol{S}$ 和 $\boldsymbol{T}$ 均为上三角阵；

② $\boldsymbol{S}$ 的对角线元素均为 -1，除第一列外，每一列另有一个非零元素 1，其余元素为零；

③ $\boldsymbol{T}$ 的第一行和对角线的元素以及其他非零元素全为 -1；

④ $\boldsymbol{S}$ 和 $\boldsymbol{T}$ 互逆。

最简单的多体系统结构为无分支的树结构，称为链结构。链结构的内接物体数满足 $i(j)=j-1$，其关联矩阵 $\boldsymbol{S}$ 的对角线右上方的次对角元素均为 1，通路矩阵 $\boldsymbol{T}$ 为充满 -1 元素的上三角阵。

2.5.3 基于拉格朗日坐标的多刚体系统运动学模型描述

目前多体系统运动学、动力学建模已经形成了比较系统的方法，其中主要有工程中常用的以拉格朗日方程为代表的分析力学方法、以牛顿-欧拉方程为代表的矢量学方法、图论方法、凯恩方法和变分方法等。目前，相关技术的软件已经得到广泛应用，如前美国机械动力公司（Mechanical Dynamics Inc.）的 ADAMS，CADSI 的 DADS，德国航天局的 SIMPACK，其它还有 WorkingModel、Flow3D、IDEAS、Phoenics、ANASYS、Pamcrash 等。

本节讨论基于拉格朗日坐标的多刚体系统运动学模型描述。拉格朗日方法是一种相对坐标方法，其特征是以多刚体系统中各铰的相对坐标为描述系统的位形坐标。以系统每个铰的一对邻接刚体为单元，以一个刚体为参考物，另一个刚体相对该刚体的位置由铰的广义坐标（拉格朗日坐标）来描述。刚体的运动可以分解为平动和定轴转动两种基本运动形式。同样，多刚体上某一点 P 的运动也由平动和转动两部分组成。开环系统的位置完全可由所有铰的拉格朗日坐标阵 $\boldsymbol{q}$ 所确定。动力学方程为拉格朗日坐标阵的二阶微分方程组

$$\boldsymbol{A}(\boldsymbol{q},t)\,\ddot{\boldsymbol{q}}=\boldsymbol{B}(\boldsymbol{q},\dot{\boldsymbol{q}},t) \tag{2.69}$$

多体系统的运动将在公共坐标系中进行考察，称为总体坐标系或总体基 $\underline{e}$。设 P_i 为多刚体上任一点，点 P_i 所在刚体部件为 B_i，$\underline{e}^i$ 是刚体 B_i 所固联的局部坐标系的基或称为连体基。

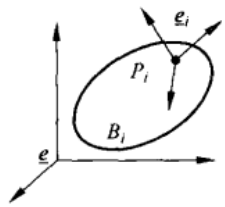


图 2.28 总体坐标系与局部坐标系

（1）刚体的姿态、角速度和角加速度

刚体 $B_i(i=1,2,\cdots,n)$ 关于系统总体基的姿态可由其关于系统总体基 $\underline{e}$ 的方向余弦阵 $\underline{\boldsymbol{A}}^i$ 来描述

$$\underline{\boldsymbol{A}}^i=\begin{cases}\underline{\boldsymbol{A}}^0\underline{\boldsymbol{A}}^{01}, & \text{当 } i=1\\ \underline{\boldsymbol{A}}^{L(i)}\underline{\boldsymbol{A}}^{L(i)i}, & \text{当 } i=2,\cdots,n\end{cases} \tag{2.70}$$

其中，$\underline{\boldsymbol{A}}^0$ 为根物体 B_0 关于系统总体基方向余弦阵，$\underline{\boldsymbol{A}}^{01}$ 为 B_1 刚体所在基 $\underline{e}^1$ 关于 B_0 刚体所在基 $\underline{e}^0$ 的方向余弦阵，$\underline{\boldsymbol{A}}^{rs}$ 表示 B_s 刚体所在基 $\underline{e}^s$ 关于 B_r 刚体所在基 $\underline{e}^r$ 的方向余弦阵。

$L(i)$ 为刚体 B_i 的内接物体的下标，$\underline{\boldsymbol{A}}^{L(i)i}$ 为 B_i 刚体所在基 $\underline{e}^i$ 关于其内接刚体 $B_{L(i)}$ 所在基 $\underline{e}^{L(i)}$ 的方向余弦阵。

刚体 $B_i(i=1,2,\cdots,n)$的绝对角速度记为 ω_i，由角速度叠加原理，刚体 B_i 的绝对角速度可以表达为如下的矢量递推公式

$$\omega_i=\begin{cases}\omega_{r1}+\omega_0, & \text{当 } i=1\\ \omega_{ri}+\omega_{L(i)}, & \text{当 } i=2,\cdots,n\end{cases}\tag{2.71}$$

式中 ω_0 为根物体的绝对角速度；ω_{r1}表示刚体 B_1 相对于其内接刚体 B_0 的角速度；ω_{ri}表示刚体 B_i 相对于其内接刚体 $B_{L(i)}$的角速度；$\omega_{L(i)}$是 B_i 的内侧刚体 $B_{L(i)}$的绝对角速度。

式(2.71)对时间求导数，刚体 $B_i(i=1,2,\cdots,n)$的绝对角加速度 $\dot{\omega}_i$ 为

$$\dot{\omega}_i=\begin{cases}\dot{\omega}_{r1}+\dot{\omega}_0, & \text{当 } i=1\\ \dot{\omega}_{ri}+\dot{\omega}_{L(i)}, & \text{当 } i=2,\cdots,n\end{cases}\tag{2.72}$$

式中 $\dot{\omega}_0$ 为根物体的绝对角加速度；同上，$\dot{\omega}_{r1}$表示刚体 B_1 相对于其内接刚体 B_0 的角加速度；$\dot{\omega}_{ri}$表示刚体 B_i 相对于其内接刚体 $B_{L(i)}$的角加速度；$\dot{\omega}_{L(i)}$是 B_i 的内侧刚体 $B_{L(i)}$的绝对角加速度。

(2) 刚体质心的位置、速度和加速度

刚体之间的相对运动既有平移也有转动，即系统既有转动铰也有滑移铰。下面首先讨论只存在相对转动的多刚体系统，然后讨论带滑移铰的多刚体系统。

① 转动铰系统刚体质心的位置、速度和加速度

首先考虑各刚体质心的位置。纯转动系统中各刚体的特点是所有与某一铰连接的物体上的各铰点重合于一点。转动铰系统中，刚体质心的位置、速度和加速度的分析与铰点在刚体上的分布密切相关，因此，首先引入描述各铰点分布情况的体铰矢量 c_{ij} 和通路矢量 d_{ij}。

如图 2.29 所示，刚体 $B_i(i=1,2,\cdots,n)$的质心记为 C_i，体铰矢量定义为由质心 C_i 指向刚体 B_i 上某铰 H_j 的矢量，记为 c_{ij}，i 为此矢量所固结刚体的下标，j 为此矢量所指向铰的下标。若铰 H_j 与刚体 B_i 不关联，$c_{ij}=0$。对于运动为已知的根物体，体铰矢量的起点 C_0 可以任选。另外，定义加权体铰矢量 $C_{ij}=S_{ij}c_{ij}$，S_{ij}为关联矩阵的元素。

对于树系统，每个物体有且只有一个内接铰，因此刚体 B_i 的通路矢量为以其内接铰 H_i 为起点的一个 B_i 的连体矢量 d_{ik}，i 为刚体 B_i 的下标，k 是 B_i 外侧的另一刚体 B_k 的下标，如图 2.30 所示。当 $k=i$ 时，该矢量由 H_i 指向 B_i 的质心 C_i，当 $k\neq i$ 时，该矢量指向 B_i 上的另一个铰——外侧铰 H_j，此铰在 B_i 至 B_k 的路上。由上有

$$d_{ik}=\begin{cases}d_{il}, & \text{当 } B_i<B_k, l=L^{-1}(i)\\ c_{ii}, & \text{当 } B_i=B_k\\ 0, & \text{其他}\end{cases}$$

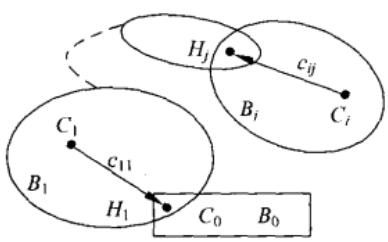


图 2.29 体铰矢量

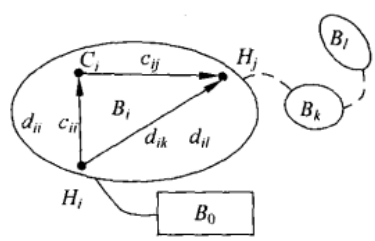


图 2.30 通路矢量

即若 B_k 和 B_l 为 B_i 的外侧且在同一路上，则 d_{ik} 和 d_{il} 为同一矢量。对于 B_i 的内侧刚体或与其不在同一路上的刚体 B_k，通路矢量 d_{ik} 为零。

体铰矢量与通路矢量均描述刚体上铰点的位置，只是矢量的起点不同而已。

设刚体质心的位置就是刚体连体基基点所在的位置，由图 2.30 知，刚体 $B_i(i=1,2,\cdots,n)$ 连体基的基点关于系统总体基的基点之矢径为

$$r_i = \sum_{k:\ B_k \in B_i} d_{ki} + r_0 \quad (i = 1,\cdots,n) \tag{2.73}$$

求和号下表示 k 为物体 B_i 至 B_0 通路上所有刚体的下标。考虑到通路矢量的性质，不在 B_i 的通路上或在 B_i 外侧的刚体 B_k 对应的广义通路矢量为零，则有

$$r_i = \sum_{k=1}^{n} d_{ki} + r_0 \quad (i = 1,\cdots,n) \tag{2.74}$$

其中，$d_{ki} = \underline{\boldsymbol{A}}^k d'_{ki}$，式中，$d'_{ki}$ 为通路矢量 d_{ki} 在连体基下的矢量形式，其分量为通路矢量在连体基下的常值坐标。因此，上式只含有刚体 B_i 至 B_0 路上的所有刚体的方向余旋矩阵 $\underline{\boldsymbol{A}}^k$ 与相应的通路矢量在连体基下的常值坐标阵。因此 B_i 的质心坐标是 B_i 至 B_0 路上的所有刚体内接铰广义坐标的函数。

各刚体质心的速度通过上式对时间进行求导而得到，有

$$\dot{r}_i = \sum_{k:\ B_k \in B_i} \dot{d}_{ki} + \dot{r}_0 \quad (i = 1,\cdots,n) \tag{2.75}$$

由于矢量 d_{ki} 固结于刚体 B_k，有 $\dot{d}_{ki} = \omega_k \times d_{ki}$，另外考虑到通路矢量的性质，因此有

$$\dot{r}_i = \sum_{k=1}^{n} \omega_k \times d_{ki} + \dot{r}_0 \quad (i = 1,\cdots,n) \tag{2.76}$$

再次求导，得到各刚体质心的加速度

$$\ddot{r}_i = \sum_{k:\ B_k \in B_i} \ddot{d}_{ki} + \ddot{r}_0 \quad (i = 1,\cdots,n) \tag{2.77}$$

由于矢量 d_{ki} 固结于刚体 B_k，有 $\ddot{d}_{ki} = \dot{\omega}_k \times d_{ki} + \omega_k \times (\omega_k \times d_{ki})$，因此

$$\ddot{r}_i = -\sum_{k=1}^{n} d_{ki} \times \dot{\omega}_k + \sum_{k:\ B_k \in B_i} \omega_k \times (\omega_k \times d_{ki}) + \ddot{r}_0 \quad (i = 1,\cdots,n) \tag{2.78}$$

② 滑移铰系统刚体质心的位置、速度和加速度

对于带滑移铰的系统，如图 2.31 中所示，H_l 为一滑移铰，h_l 为滑移铰 H_l 的移动铰矢量。B_k 为 B_l 的内接刚体，即有 $k = L^+(l)$，或 $l = L^{-1}(k)$。描述系统刚体的运动，需要定义广义通路矢量 d^*_{ki}：当 $k \neq i$，且 B_i 在 B_k 的外侧时，该矢量指向其外接刚体 B_l 的内接铰点；B_i 在 B_k 的内侧或 B_i 与 B_k 不在同一条通路时该矢量为零；当 $k=i$ 时，则与通路矢量的定义一致，该矢量指向 B_k 的质心。引入通路元素，考虑到 $T_{ii} = -1$，则上述定义可表示为

$$d^*_{ki} = \begin{cases} d_{ki}, & \text{当 } B_k < B_i, l = L^{-1}(k) \\ d_{kk}, & \text{当 } B_k = B_i \\ 0, & \text{其他} \end{cases}$$

这里需要注意的是，通路矢量 d_{ki} 是一个固结于刚体 B_k 的矢量，因此是一个常矢量。然而当 $k \neq i$ 时，由于滑移铰 H_l 的

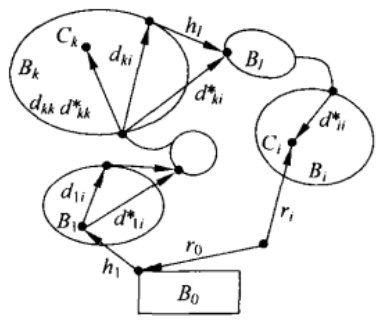


图 2.31 广义通路矢量

移动铰矢量 h_l 是移动的，广义通路矢量却是相对于刚体 B_k 的一个动矢量。当 H_l 退化为一个转动铰时，该广义通路矢量 d_{ki}^* 退化为通路矢量 d_{ki}。

下面考虑刚体质心的位置。由图 2.31 知，刚体 B_i 的质心关于系统总体基的基点的矢径为

$$r_i = \sum_{k:\ B_k \in B_i} d_{ki}^* + r_0 + h_1 \quad (i = 1,2,\cdots,n) \tag{2.79}$$

求和号下表示 k 为物体 B_i 至 B_0 通路上所有刚体的下标。考虑到通路矢量的性质，不在 B_i 的通路上或在 B_i 外侧的刚体 B_k 对应的广义通路矢量为零，则有

$$r_i = \sum_{k=1}^{n} d_{ki}^* + r_0 + h_1 \quad (i = 1,2,\cdots,n) \tag{2.80}$$

考虑上面 d_{ki} 式中的第一个分式，$d_{ki}^* = d_{ki} - T_{li} h_l$，则有

$$\dot{d}_{ki}^* = \dot{d}_{ki} - T_{li}\dot{h}_l$$

而

$$\dot{h}_l = v_{rl} + \omega_k \times h_l$$

式中，v_{rl} 为铰 H_l 的相对移动速度。因此

$$\dot{d}_{ki}^* = \omega_k \times d_{ki}^* - T_{li} v_{rl} \tag{2.81}$$

将式(2.80)在总体基对时间求导，并根据通路矢量的性质，可得到各刚体质心的速度

$$\dot{r}_i = \sum_{k=1}^{n} \omega_k \times d_{ki}^* - \sum_{l=1}^{n} T_{li} v_{rl} + v_{r1} + \omega_0 \times h_1 + \dot{r}_0 \quad (i = 1,\cdots,n;\ l = L^{\dagger}(k)) \tag{2.82}$$

右边第二项的物理意义在刚体 B_l 至 B_i 的路上所有铰的相对移动速度的矢量和，加上第三项，则为刚体 B_0 至 B_i 的路上所有铰相对移动速度的矢量和。

将式(2.82)求导，由于 $\ddot{d}_{ki}^* = \dot{\omega}_k \times d_{ki}^* + \omega_k \times \dot{d}_{ki}^* - T_{li}(\bar{v}_{rl} + \omega_k \times v_{rl})$

有

$$\ddot{r}_i = \sum_{k=1}^{n} \dot{\omega}_k \times d_{ki}^* - 2\sum_{k=1}^{n} T_{li}\omega_k \times v_{rl} + \sum_{k=1}^{n} \omega_k \times (\omega_k \times d_{ki}^*) - \sum_{k=1}^{n} T_{li}\bar{v}_{rl} + \bar{v}_{r1}$$
$$+ 2\omega_0 \times v_{r1} + \dot{\omega}_0 \times h_1 + \omega_0 \times (\omega_0 \times h_1) + \ddot{r}_0 \quad (i = 1,\cdots,n) \tag{2.83}$$

当系统只有转动铰而无滑移铰时，广义通路矢量退化为通路矢量，相对移动的矢量和相对速度项均为零，这时，与转动铰系统一致。

由以上刚体在绝对坐标系下位置、速度和加速度的表达式，结合广义坐标与刚体绝对坐标之间的关系，可以进一步推导出广义坐标下，刚体位置、速度和加速度的表达式。

对于无根树系统，通常需讨论系统相对于某个运动为已知的动参考基的运动。如对于航天器，可取轨道坐标为动基。为了让系统的拓扑构型在运动学上与有根系统取得一致，令这个动参考基为 B_0。将系统中某一刚体设定为 B_1，与 B_0 以虚铰方式连接，如图 2.32 所示。虚铰 H_1 对于邻接刚体 B_1 和 B_0 无任何运动学约束，也不存在约束力和力矩。虚铰 H_1 的位置也可以任意设定，为了方便将它的一端取在动参考基 B_0 的基点，另一端与 B_1 的质心 C_1 重合，这样，体铰矢量 $c_{11} = 0$。

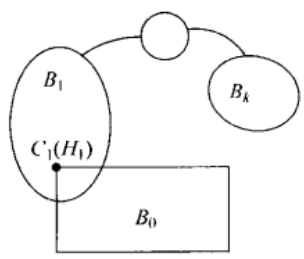


图 2.32 无根树系统

由于在系统的拓扑构型与运动学上，上述定义的无根系统与一个 H_1 为六自由度的有根系统等价，因此各刚体的位姿、绝对角速度与角加速度的公式与前述有根系统的相同。各刚体质心的位置、绝对速度和绝对加速度与带滑移铰系统的公式相同。

2.5.4 基于 Haug 笛卡儿方法的多刚体系统运动学模型描述

20 世纪 80 年代，美国 Chance 和 Haug 提出了一种适于计算机辅助建模与求解的多刚体系统笛卡儿建模方法。它以系统中每一个物体为单元，建立固结在刚体上的坐标系，刚体的坐标相对于一个公共参考基进行定义，其位置坐标统一为刚体坐标系基点的笛卡儿坐标与坐标系的方位坐标，再根据铰约束和动力学原理建立系统模型。

对于由 N 个刚体组成的系统，位置坐标阵 $\boldsymbol{q}$ 中的坐标个数为 $3N$ 个(二维)或 $6N$(如果采用欧拉参数为 $7N$)个(三维)。由于存在铰约束，位置坐标不独立。动力学模型形式为

$$\begin{cases} \boldsymbol{A}\ddot{\boldsymbol{q}} + \boldsymbol{\Phi}_{\mathrm{q}}^{\mathrm{T}}\lambda = \boldsymbol{B} \\ \boldsymbol{\Phi}(\boldsymbol{q},t) = 0 \end{cases} \tag{2.84}$$

其中 $\boldsymbol{\Phi}$ 为位置坐标阵 $\boldsymbol{q}$ 的约束方程，$\boldsymbol{\Phi}_{\mathrm{q}}$ 为约束方程的雅可比矩阵，λ 为拉格朗日乘子。

设系统由 nb 个刚性构件组成，先考虑二维情况，建立全局坐标系 xoy，以表示系统的运动；再构建每个构件 i 的连体坐标系 $x_i'o_i'y_i'$，以表示该构件在连体坐标系的运动。

构件 i 的连体坐标系原点 o_i' 的全局坐标用 r_i 表示：$r_i=[x_i,y_i]^{\mathrm{T}}$，连体坐标系 $x_i'o_i'y_i'$ 相对于全局坐标系 xoy 的转角为 ϕ_i，则笛卡儿广义坐标矢量为 $q_i\equiv[x_i,y_i,\phi_i]^{\mathrm{T}}$。

由 nb 个刚性构件组成的系统广义坐标数为 $nc=3nb$，系统广义坐标为 $\boldsymbol{q}=[q_1^{\mathrm{T}},q_2^{\mathrm{T}},\cdots,q_{nb}^{\mathrm{T}}]^{\mathrm{T}}$。

这样，系统中的运动副连接可以用广义坐标表示为代数方程，亦称为约束方程，设为 nh 个，用下式表示

$$\boldsymbol{\Phi}^K(\boldsymbol{q}) = [\Phi_1^K(\boldsymbol{q}),\Phi_2^K(\boldsymbol{q}),\cdots,\Phi_{nh}^K(\boldsymbol{q})]^{\mathrm{T}} = 0 \tag{2.85}$$

这是定常完整约束，还有非定常约束(如与时间有关)、非完整约束(如不等式约束)。若 $nc>nh$，且 nh 个方程是独立、相容的，则称系统具有 $DOF=nc-nh$ 个自由度。为使系统具有确定运动，往往采取：

① 添加与系统自由度相等的附加驱动约束，此时，系统自由度为 0，被称为运动学上确定的，求解系统则是运动学分析，不涉及力或反作用力。但可求约束反力，即已知运动求作用力，称为动力学逆问题。

② 对系统施加力，此时，系统自由度大于 0，但有外力作用，求解系统则是动力学分析。如果力与时间无关，则是静平衡分析问题。

(1) 运动学分析

对运动学分析，施加 $nc-nh$ 个驱动约束 $\boldsymbol{\Phi}^D(\boldsymbol{q},t)=0$，从而全部约束表示为

$$\boldsymbol{\Phi}(\boldsymbol{q},t) = \begin{bmatrix} \boldsymbol{\Phi}^K(\boldsymbol{q},t) \\ \boldsymbol{\Phi}^D(\boldsymbol{q},t) \end{bmatrix} = 0 \tag{2.86}$$

这就是位置方程，求解可得广义坐标位置 $\boldsymbol{q}(t)$。相应地，可得速度与加速度方程

$$\dot{\boldsymbol{\Phi}}(\boldsymbol{q},\dot{\boldsymbol{q}},t) = \boldsymbol{\Phi}_{\mathrm{q}}(\boldsymbol{q},t)\,\dot{\boldsymbol{q}} + \boldsymbol{\Phi}_{\mathrm{t}}(\boldsymbol{q},t) = 0 \tag{2.87}$$

$$\ddot{\boldsymbol{\Phi}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}},t)=\boldsymbol{\Phi}_{\mathrm{q}}(\boldsymbol{q},t)\,\ddot{\boldsymbol{q}}+(\boldsymbol{\Phi}_{\mathrm{q}}(\boldsymbol{q},t)\,\dot{\boldsymbol{q}})_{\mathrm{q}}\,\dot{\boldsymbol{q}}+2\boldsymbol{\Phi}_{\mathrm{qt}}(\boldsymbol{q},t)\,\dot{\boldsymbol{q}}+\boldsymbol{\Phi}_{\mathrm{tt}}(\boldsymbol{q},t)=0 \quad (2.88)$$

令$\boldsymbol{v}(\boldsymbol{q},t)=-\boldsymbol{\Phi}_{\mathrm{t}}(\boldsymbol{q},t)$，$\boldsymbol{\eta}(\boldsymbol{q},\dot{\boldsymbol{q}},t)=-(\boldsymbol{\Phi}_{\mathrm{q}}(\boldsymbol{q},t)\dot{\boldsymbol{q}})_{\mathrm{q}}\,\dot{\boldsymbol{q}}-2\boldsymbol{\Phi}_{\mathrm{qt}}(\boldsymbol{q},t)\dot{\boldsymbol{q}}+\boldsymbol{\Phi}_{\mathrm{tt}}(\boldsymbol{q},t)$，则可得标准表达式

$$\begin{cases}\boldsymbol{\Phi}(\boldsymbol{q},t)=0\\ \boldsymbol{\Phi}_{\mathrm{q}}(\boldsymbol{q},t)\,\dot{\boldsymbol{q}}=\boldsymbol{v}(\boldsymbol{q},t)\\ \boldsymbol{\Phi}_{\mathrm{q}}(\boldsymbol{q},t)\,\ddot{\boldsymbol{q}}=\boldsymbol{\eta}(\boldsymbol{q},\dot{\boldsymbol{q}},t)\end{cases} \quad (2.89)$$

关键是雅可比矩阵$\boldsymbol{\Phi}_{\mathrm{q}}(\boldsymbol{q},t)$必须非奇异，这样就可求解。如果$\boldsymbol{\Phi}$的维数为$m$，$\boldsymbol{q}$的维数为$nb$，则$\boldsymbol{\Phi}_{\mathrm{q}}$的维数为$m\times nb$。

为了得到全局坐标系的值，必须进行坐标变换。设矢量$\boldsymbol{s}$在全局坐标系xOy和某连体坐标系$x'O'y'$中分别表示为(参见图2.33)

$$\begin{cases}\boldsymbol{s}=[s_x,s_y]^{\mathrm{T}}\\ \boldsymbol{s}'=[s_{x'},s_{y'}]^{\mathrm{T}}\end{cases}$$

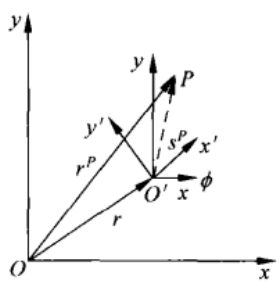


图2.33 坐标系示意

任意点P的坐标变换关系为

$$r^P=r+s^P=r+\boldsymbol{A}s'^P \quad (2.90)$$

其中 r^P——P在全局坐标系中的坐标；

r——连体坐标系原点O'在全局坐标系中的坐标；

s^P——$\boldsymbol{s}$在全局坐标系中的坐标；

s'^P——$\boldsymbol{s}$在连体坐标系中的坐标。

$\boldsymbol{A}$为旋转变换矩阵

$$\boldsymbol{A}=\boldsymbol{A}(\phi)=\begin{bmatrix}\cos\phi & -\sin\phi\\ \sin\phi & \cos\phi\end{bmatrix}$$

$\boldsymbol{A}$对时间的导数

$$\dot{\boldsymbol{A}}=\dot{\phi}\frac{\mathrm{d}}{\mathrm{d}\phi}\boldsymbol{A}=\dot{\phi}\begin{bmatrix}-\sin\phi & -\cos\phi\\ \cos\phi & -\sin\phi\end{bmatrix}=\dot{\phi}\boldsymbol{B}$$

速度变换公式

$$\dot{r}^P=\dot{r}+\dot{\boldsymbol{A}}s'^P=\dot{r}+\dot{\phi}\boldsymbol{B}s'^P \quad (2.91)$$

加速度变换公式

$$\ddot{r}^P=\ddot{r}+\ddot{\phi}\boldsymbol{B}s'^P-\dot{\phi}^2\boldsymbol{A}s'^P \quad (2.92)$$

对三维情况，表达式复杂一些

方向余弦矩阵

$$\boldsymbol{A}=[\boldsymbol{f},\boldsymbol{g},\boldsymbol{h}]=\begin{bmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix}$$

$\boldsymbol{f},\boldsymbol{g},\boldsymbol{h}$分别为连体坐标系$O'x'y'z'$三个坐标轴的单位矢量，$\boldsymbol{A}$为正交矩阵。

任意点P的坐标变换关系为

$$r^P=r+s^P+\boldsymbol{A}s'^P$$

速度变换公式

$$\dot{r}^P = \dot{r} + \dot{A}s'^P = \dot{r} + \tilde{\omega}s'^P$$

加速度变换公式

$$\ddot{r}^P = \ddot{r} + \ddot{A}s'^P = \ddot{r} + \tilde{\dot{\omega}}As'^P + \tilde{\omega}\tilde{\omega}As'^P$$

其中，$\tilde{\omega}$ 是 ω 的斜对称矩阵，ω 称为连体坐标系 $O'x'y'z'$ 对于全局坐标系 $Oxyz$ 的角速度矢量，且 $\tilde{\omega} = \dot{A}A^{\mathrm{T}}$。

(2) 与欧拉参数的关系

如果连体坐标系 $O'x'y'z'$ 与全局坐标系 $Oxyz$ 原点重合，设 $Oxyz$ 绕单位轴矢量 $\vec{u}$ 转动 χ 角与坐标系 $O'x'y'z'$ 重合，定义欧拉参数组

$$\begin{cases} e_0 = \cos\dfrac{\chi}{2} \\ e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \equiv u\sin\dfrac{\chi}{2} \end{cases} \tag{2.93}$$

用向量表示

$$p = [e_0, e^{\mathrm{T}}]^{\mathrm{T}} = [e_0, e_1, e_2, e_3]^{\mathrm{T}}$$

注意要满足归一化约束

$$p^{\mathrm{T}}p = e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1$$

则

$$A = (2e_0^2 - 1)I + 2(ee^{\mathrm{T}} + e_0\tilde{e}) \tag{2.94}$$

其中 I 为 3×3 单位阵，$\tilde{e}$ 为 e 的斜对称矩阵(斜方阵)

$$\tilde{e} = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix} \tag{2.95}$$

刚体的方位从连体坐标系与全局坐标系重合的始点起依次绕 $O'x'y'z'$ 的三个轴转过进动角 ψ、章动角 θ、自转角 ϕ(这三个角度即为欧拉角)，则用欧拉角表示方向余弦为

$$A = \begin{bmatrix} c\psi \cdot c\phi - s\psi \cdot c\theta \cdot s\phi & -c\psi \cdot s\phi - s\psi \cdot c\theta \cdot c\phi & s\psi \cdot s\theta \\ s\psi \cdot c\phi + c\psi \cdot c\theta \cdot s\phi & -s\psi \cdot s\phi + c\psi \cdot c\theta \cdot c\phi & -c\psi \cdot s\theta \\ s\theta \cdot s\phi & s\theta \cdot c\phi & c\theta \end{bmatrix}$$

欧拉角与欧拉参数的关系式为

$$\begin{cases} e_0 = \cos\left(\dfrac{\psi+\phi}{2}\right)\cos\left(\dfrac{\theta}{2}\right) \\ e_1 = \cos\left(\dfrac{\psi-\phi}{2}\right)\sin\left(\dfrac{\theta}{2}\right) \\ e_2 = \sin\left(\dfrac{\psi-\phi}{2}\right)\sin\left(\dfrac{\theta}{2}\right) \\ e_3 = \sin\left(\dfrac{\psi+\phi}{2}\right)\cos\left(\dfrac{\theta}{2}\right) \end{cases} \tag{2.96}$$

为了进行坐标变换，由构件 i 的广义坐标矢量表示为连体坐标系原点和欧拉位置参数

$$q_i = \begin{bmatrix} r_i \\ p_i \end{bmatrix}, \quad \boldsymbol{q} = [q_1^{\mathrm{T}}, q_2^{\mathrm{T}}, \cdots, q_{nb}^{\mathrm{T}}]^{\mathrm{T}}$$

维数为 $7nb$，注意必须满足归一化约束 $\Phi_i^p = p_i^{\mathrm{T}} p_i - 1 = 0 (i = 1, 2, \cdots, nb)$。从而

参数归一化约束方程

$$\boldsymbol{\Phi}^p = [\Phi_1^p, \Phi_2^p, \cdots, \Phi_{nb}^p]^{\mathrm{T}} = 0 \qquad (nb \text{ 维})$$

与运动副等价的约束方程

$$\boldsymbol{\Phi}^K(\boldsymbol{q}) = [\Phi_1^K(\boldsymbol{q}), \Phi_2^K(\boldsymbol{q}), \cdots, \Phi_{nh}^K(\boldsymbol{q})]^{\mathrm{T}} = 0 \qquad (nh \text{ 维})$$

驱动约束方程

$$\boldsymbol{\Phi}^D(\boldsymbol{q}, t) = 0 \qquad (6nb - nh \text{ 维})$$

从而得到

位置方程

$$\boldsymbol{\Phi}(\boldsymbol{q}, t) \equiv \begin{bmatrix} \boldsymbol{\Phi}^p(\boldsymbol{q}) \\ \boldsymbol{\Phi}^K(\boldsymbol{q}) \\ \boldsymbol{\Phi}^D(\boldsymbol{q}, t) \end{bmatrix} = 0 \qquad (7nb \text{ 个方程})$$

速度方程

$$\sum_{i=1}^{nb} \begin{bmatrix} \Phi_{ri}^K \\ \Phi_{ri}^D \end{bmatrix} \dot{r}_i + \begin{bmatrix} \Phi_{\pi' i}^K \\ \Phi_{\pi' i}^D \end{bmatrix} \omega_i' = \begin{bmatrix} -\boldsymbol{\Phi}_t^K \\ -\boldsymbol{\Phi}_t^D \end{bmatrix} \equiv \begin{bmatrix} \boldsymbol{v}^K \\ \boldsymbol{v}^D \end{bmatrix}$$

加速度方程

$$\sum_{i=1}^{nb} \left\{ \begin{bmatrix} \Phi_{ri}^K \\ \Phi_{ri}^D \end{bmatrix} \ddot{r}_i + \begin{bmatrix} \Phi_{\pi' i}^K \\ \Phi_{\pi' i}^D \end{bmatrix} \dot{\omega}_i' \right\} = -\begin{bmatrix} \boldsymbol{\Phi}_{tt}^K \\ \boldsymbol{\Phi}_{tt}^D \end{bmatrix} - \sum_{i=1}^{nb} \left\{ \begin{bmatrix} \dot{\Phi}_{ri}^K \\ \dot{\Phi}_{ri}^D \end{bmatrix} \dot{r}_i + \begin{bmatrix} \dot{\Phi}_{\pi' i}^K \\ \dot{\Phi}_{\pi' i}^D \end{bmatrix} \omega_i' \right\} = \begin{bmatrix} \boldsymbol{\eta}^K \\ \boldsymbol{\eta}^D \end{bmatrix}$$

注意，$\boldsymbol{\pi}'$ 是通过变分得到的，满足如下关系

$$\delta \boldsymbol{\pi}' = 2[-\boldsymbol{e}, -\tilde{\boldsymbol{e}} + e_0 \boldsymbol{I}] \delta p$$

即

$$\delta \boldsymbol{\pi}' = 2 \begin{bmatrix} -e_1 & e_0 & e_3 & -e_2 \\ -e_2 & -e_3 & e_0 & e_1 \\ -e_3 & e_2 & -e_1 & e_0 \end{bmatrix} \delta p \tag{2.97}$$

2.5.5 多刚体系统动力学模型描述

对于受约束的多体系统，其动力学方程建立包括以下内容：

① 根据牛顿定理给出物体的变分运动方程；

② 运用拉格朗日乘子定理导出基于约束的动力学方程。

1. 自由物体变分运动方程

以二维为例，刚体构件 i 的质量为 m_i，对质心的极转动惯量为 J_i'，作用于刚体的所有外力向质心简化得到外力矢量 $\boldsymbol{F}_i$ 和力矩 $\boldsymbol{n}_i$。根据牛顿定理，给出构件 i 的变分运动方程为

$$\delta \boldsymbol{r}_i^{\mathrm{T}} [m_i \ddot{\boldsymbol{r}}_i - \boldsymbol{F}_i] + \delta \phi_i [J_i' \ddot{\phi}_i - \boldsymbol{n}_i] = 0 \tag{2.98}$$

其中 $\boldsymbol{r}_i$ 为固定位于刚体质心的连体坐标系原点 o' 的代数矢量，ϕ_i 为连体坐标系相对全局坐标系的转角，$\delta\boldsymbol{r}_i$ 与 $\delta\phi_i$ 分别为 $\boldsymbol{r}_i$、ϕ_i 的变分。

构件 i 的广义坐标为 $\boldsymbol{q}_i=[\boldsymbol{r}_i^{\mathrm{T}},\phi_i]^{\mathrm{T}}$，定义广义力 $\boldsymbol{Q}_i=[\boldsymbol{F}_i^{\mathrm{T}},\boldsymbol{n}_i]^{\mathrm{T}}$，质量矩阵 $\boldsymbol{M}_i=\mathrm{diag}(m_i,m_i,J_i')$，从而式(2.98)写成虚功原理形式

$$\delta\boldsymbol{q}_i^{\mathrm{T}}(\boldsymbol{M}_i\ddot{\boldsymbol{q}}_i-\boldsymbol{Q}_i)=0 \tag{2.99}$$

下面再讨论约束多体系统运动方程。系统由 nb 个构件组成，分别运用式(2.99)，组合后可得到其矩阵形式

$$\delta\boldsymbol{q}^{\mathrm{T}}[\boldsymbol{M}\ddot{\boldsymbol{q}}-\boldsymbol{Q}]=0 \tag{2.100}$$

其中 $\boldsymbol{q}=[\boldsymbol{q}_1^{\mathrm{T}},\boldsymbol{q}_2^{\mathrm{T}},\cdots,\boldsymbol{q}_{nb}^{\mathrm{T}}]^{\mathrm{T}}$，$\boldsymbol{M}=\mathrm{diag}[\boldsymbol{M}_1,\boldsymbol{M}_2,\cdots,\boldsymbol{M}_{nb}]$，$\boldsymbol{Q}=[\boldsymbol{Q}_1^{\mathrm{T}},\boldsymbol{Q}_2^{\mathrm{T}},\cdots,\boldsymbol{Q}_{nb}^{\mathrm{T}}]^{\mathrm{T}}$。

根据牛顿第三定律，考虑到广义力中的约束力总虚功为 0，外加作用力表示为

$\boldsymbol{Q}^A=[\boldsymbol{Q}_1^{A^{\mathrm{T}}},\boldsymbol{Q}_2^{A^{\mathrm{T}}},\cdots,\boldsymbol{Q}_{nb}^{A^{\mathrm{T}}}]^{\mathrm{T}}$，其中 $\boldsymbol{Q}_i^A=[\boldsymbol{F}_i^{\mathrm{T}^A},\boldsymbol{n}^A]^{\mathrm{T}}$，$i=1,2,\cdots,nb$。

注意到，运动学约束与驱动约束为 $\boldsymbol{\Phi}(\boldsymbol{q},t)$，同样可得到其变分形式

$$\boldsymbol{\Phi}_{\mathrm{q}}\delta\boldsymbol{q}=0 \tag{2.101}$$

式(2.100)、式(2.101)构成受约束的多体系统变分运动方程。

2. 拉格朗日乘子

定理：设矢量 $\boldsymbol{b}\in R^n$，矢量 $\boldsymbol{x}\in R^n$，矩阵 $\boldsymbol{A}\in R^{m\times n}$ 为常数阵，如果使得 $\boldsymbol{A}\boldsymbol{x}=0$ 成立的 x 都能有 $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{x}=0$，则存在拉格朗日乘子矢量 $\boldsymbol{\lambda}\in R^m$，满足 $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{x}+\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}=0$。

对式(2.100)、式(2.101)运用拉格朗日乘子可得

$$[\boldsymbol{M}\ddot{\boldsymbol{q}}-\boldsymbol{Q}^A]^{\mathrm{T}}\delta\boldsymbol{q}+\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{\Phi}_{\mathrm{q}}\delta\boldsymbol{q}=[\boldsymbol{M}\ddot{\boldsymbol{q}}+\boldsymbol{\Phi}_{\mathrm{q}}^{\mathrm{T}}\boldsymbol{\lambda}-\boldsymbol{Q}^A]^{\mathrm{T}}\delta\boldsymbol{q}=0$$

从而得到动力学方程

$$\boldsymbol{M}\ddot{\boldsymbol{q}}+\boldsymbol{\Phi}_{\mathrm{q}}^{\mathrm{T}}\boldsymbol{\lambda}=\boldsymbol{Q}^A \tag{2.102}$$

称为欧拉-拉格朗日方程。注意式(2.102)满足如下位置约束、速度约束、加速度约束方程

$$\begin{cases}\boldsymbol{\Phi}(\boldsymbol{q},t)=0\\ \dot{\boldsymbol{\Phi}}(\boldsymbol{q},\dot{\boldsymbol{q}},t)=\boldsymbol{\Phi}_{\mathrm{q}}(\boldsymbol{q},t)\dot{\boldsymbol{q}}-\boldsymbol{v}(\boldsymbol{q},t)=0,\boldsymbol{v}=-\boldsymbol{\Phi}_{\mathrm{t}}(\boldsymbol{q},t)\\ \ddot{\boldsymbol{\Phi}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}},t)=\boldsymbol{\Phi}_{\mathrm{q}}(\boldsymbol{q},t)\ddot{\boldsymbol{q}}-\boldsymbol{\eta}(\boldsymbol{q},\dot{\boldsymbol{q}},t)=0,\boldsymbol{\eta}=-(\boldsymbol{\Phi}_{\mathrm{q}}\dot{\boldsymbol{q}})_{\mathrm{q}}-2\boldsymbol{\Phi}_{\mathrm{qt}}\dot{\boldsymbol{q}}-\boldsymbol{\Phi}_{\mathrm{tt}}\end{cases} \tag{2.103}$$

矩阵形式

$$\begin{bmatrix}\boldsymbol{M} & \boldsymbol{\Phi}_{\mathrm{q}}^{\mathrm{T}}\\ \boldsymbol{\Phi}_{\mathrm{q}} & 0\end{bmatrix}\begin{bmatrix}\ddot{\boldsymbol{q}}\\ \boldsymbol{\lambda}\end{bmatrix}=\begin{bmatrix}\boldsymbol{Q}^A\\ \boldsymbol{\eta}\end{bmatrix} \tag{2.104}$$

求解的条件是系数矩阵非奇异。实际上只要保证约束是独立的，方程就有解；同时要保证初值满足相容的条件，即位置约束方程为 0，速度约束方程为 0。

① 正向动力学分析：若系统约束方程的维数 m 小于广义坐标维数 n，则施加外力直接求解，得到位置、速度、加速度以及代表约束反力的拉格朗日乘子。

② 逆向动力学分析：若系统约束方程的维数 m 等于广义坐标维数 n，系统在运动学上完全确定，先计算 $\ddot{\boldsymbol{q}}$，然后计算 $\boldsymbol{\lambda}$，从而得到约束反力。

③ 静平衡分析：如果系统在外力作用下保持静止状态，即 $\ddot{\boldsymbol{q}}=\dot{\boldsymbol{q}}=0$，则式(2.102)退化为

$$\boldsymbol{\Phi}_{\mathrm{q}}^{\mathrm{T}}\boldsymbol{\lambda}=\boldsymbol{Q}^A \tag{2.105}$$

将其与位置约束方程联立求解得到状态 $\boldsymbol{q}$ 与拉格朗日乘子 $\boldsymbol{\lambda}$ 。

对于三维情况，刚体构件 i 的质量为 m_i，对质心的惯量张量为 J_i'，作用于刚体的所有外力向质心简化得到外力矢量 $\boldsymbol{F}_i$ 和力矩 $\boldsymbol{n}_i'$。这里只给出表达式，推导过程要复杂得多。

带约束多体系统变分运动方程

$$\delta \boldsymbol{r}^{\mathrm{T}}[\boldsymbol{M}\ddot{\boldsymbol{r}} - \boldsymbol{F}^{A}] + \delta \boldsymbol{\pi}'^{\mathrm{T}}[\boldsymbol{J}'\dot{\boldsymbol{\omega}}' + \tilde{\boldsymbol{\omega}}'\boldsymbol{J}'\boldsymbol{\omega}' - \boldsymbol{n}'^{A}] = 0 \tag{2.106}$$

运动学和驱动约束为

$$\boldsymbol{\Phi}(\boldsymbol{r},\boldsymbol{p},t) \equiv \begin{bmatrix} \boldsymbol{\Phi}^{K}(\boldsymbol{r},\boldsymbol{p}) \\ \boldsymbol{\Phi}^{D}(\boldsymbol{r},\boldsymbol{p},t) \end{bmatrix} = 0 \tag{2.107}$$

应用拉格朗日定理后得到

$$\boldsymbol{M}\ddot{\boldsymbol{r}} + \boldsymbol{\Phi}_{r}^{\mathrm{T}}\boldsymbol{\lambda} = \boldsymbol{F}^{A}$$

$$\boldsymbol{J}'\dot{\boldsymbol{\omega}}' + \boldsymbol{\Phi}_{\pi'}^{\mathrm{T}}\boldsymbol{\lambda} = \boldsymbol{n}'^{A} - \tilde{\boldsymbol{\omega}}'\boldsymbol{J}'\boldsymbol{\omega}'$$

方阵形式

$$\begin{bmatrix} \boldsymbol{M} & 0 & \boldsymbol{\Phi}_{r}^{\mathrm{T}} \\ 0 & \boldsymbol{J}' & \boldsymbol{\Phi}_{\pi'}^{\mathrm{T}} \\ \boldsymbol{\Phi}_{R} & \boldsymbol{\Phi}_{\pi'} & 0 \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{r}} \\ \dot{\boldsymbol{\omega}}' \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}^{A} \\ \boldsymbol{n}'^{A} - \tilde{\boldsymbol{\omega}}'\boldsymbol{J}'\boldsymbol{\omega}' \\ \boldsymbol{\eta} \end{bmatrix} \tag{2.108}$$

2.5.6 小结

系统运动和动力学模型是多体系统仿真的基础。工程领域中的机械系统是由大量零部件构成的，传统的以经典力学为依托的分析方法已不能应付这些复杂工程对象的动力学问题，必须借助于先进的计算机辅助模型设计。

本节讨论了基于图论的多体系统运动学、动力学建模方法学，包括基于图论建立多体系统的拓扑结构，在此基础上，介绍了两种运动学建模方法，其一是基于拉格朗日坐标的多刚体系统运动学模型，其二是基于 Haug 笛卡儿方法的多刚体系统运动学模型。还简单讨论了基于变分法的多刚体动力学建模方法。这些模型的描述形式非常适于计算机处理，从而为计算机辅助模型设计提供了有效手段。

当然，实际系统可能还包括柔性体，甚至塑性体，这类系统的建模就更为复杂，本书限于篇幅不做进一步介绍，有兴趣的读者可参考相关的专业文献和书籍。

习题

1. 描述系统的微分方程为 $5\dddot{y}(t)+3\ddot{y}(t)+2\dot{y}(t)=\ddot{u}(t)+2\dot{u}(t)+3u(t)$，已知 y,u 及其各阶导数初始值为零，试将其转换成状态空间表达式。

2. 系统传递函数为

$$G(s) = \frac{Y(s)}{U(s)} = \frac{s^2+3s+2}{s(s^2+7s+12)}$$

已知 y,u 及其各阶导数初始值为零，试将其转换成系统状态空间表达式。

3. 用书中的伴随方程法将 $3\ddot{y}+5\dot{y}+20y=3\dot{u}+u$ 转换成状态方程，并求出对应状态变量的初值。已知 $y_0=2.0$，$\dot{y}_0=-1.0$，$u_0=1$，$\dot{u}_0=0.5$。

4. 写出如图 2.34 所示系统的联结矩阵 w_1 及输入矩阵 w_0，其中方框内的数字表示环节的编号。

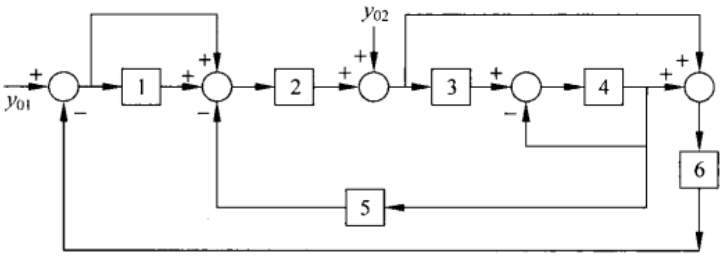


图 2.34 习题 4 系统结构图

5. 试用结构变换程序法，对如图 2.35 所示模型进行变换，导出其微分方程和代数方程。

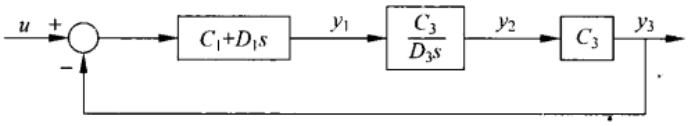


图 2.35 习题 5 系统结构图

6. 试用微分方程 $\sum_{i=0}^{n}a_ip^iy=\sum_{i=0}^{n}b_ip^iu$（其中 p 为微分算子），推导下列伴随方程的转换公式

$$y=x_1+c_nu$$

$$\dot{x}_i=x_{i+1}+c_{n-i}u \qquad (i=1,2,\cdots,n-1)$$

$$\dot{x}_n=-\frac{1}{a_n}\left(\sum_{i=1}^{n}a_{i-1}x_i\right)+c_0u$$

当 $a_n=1$ 时，$c_n=b_n$，$c_i=b_i-\sum_{j=i}^{n-1}a_jc_{n-j+i}$，且初值转换公式为

$$x_{i0}=y_0^{(i-1)}-\sum_{j=0}^{i-1}c_{n\ i+j+1}u_0^{(j)}$$

7. 在人口增长因素的分析模型中，考虑以下 12 个因素：

(1) 期望寿命 s_1　　(7) 环境污染 s_7

(2) 医疗保健水平 s_2　　(8) 国民收入 s_8

(3) 国民生育能力 s_3　　(9) 国民素质 s_9

(4) 计划生育政策 s_4　　(10) 出生率 s_{10}

(5) 国民思想、风俗 s_5　　(11) 死亡率 s_{11}

(6) 环境污染程度 s_6　　(12) 总人口 s_{12}

它们之间的相互影响关系用一个倒半三角表示，其中第 i 行从左到右分别表示 s_{12}，s_{11}，…同 s_i 的关系，其中 v 表示 s_i 影响其它因素，a 表示其它因素影响 s_i，va 表示 s_i 同其它因素相互影响。请画出人口增长因素分析的结构模型。（答案参见图 2.36）

	s_{12}	s_{11}	s_{10}	s_9	s_8	s_7	s_6	s_5	s_4	s_3	s_2	s_1
s_1	v	v	0	0	a	a	a	0	0	0	a	0
s_2	v	v	0	0	0	0	0	0	0	v	0	
s_3	v	0	v	0	a	0	a	0	0	0		
s_4	v	0	v	a	a	0	0	va	0			
s_5	v	0	v	a	a	0	0	0				
s_6	v	v	v	0	0	0	0					
s_7	v	v	v	0	0	0						
s_8	v	v	v	0	0							
s_9	v	0	v	0								
s_{10}	v	0	0									
s_{11}	v	0										
s_{12}	0											

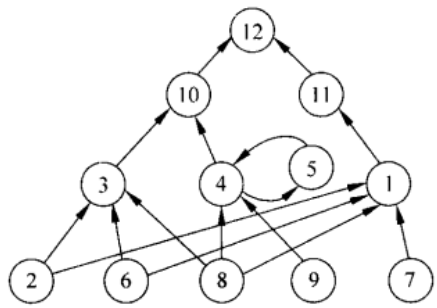


图 2.36 人口增长因素分析结构模型设计

8. 用节点法列出图 2.37 所示的电路的方程，并给出电路的关联矩阵。

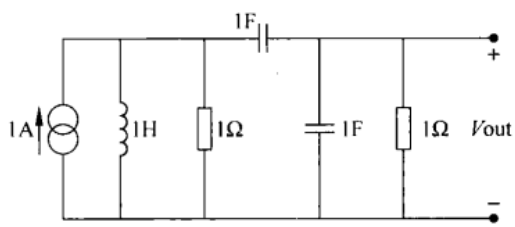


图 2.37 习题 8 的电路图

9. 用改进节点法列出图 2.38 所示的电路方程。

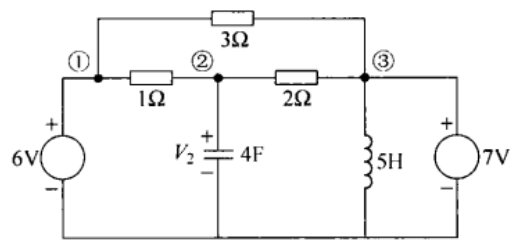


图 2.38 习题 9 的电路图

10. 用改进节点法列出图 2.39 所示的电路方程。以阻抗的形式输入电感，且处理初始条件。

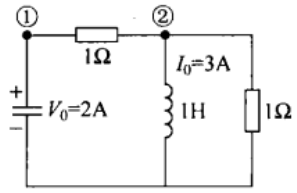


图 2.39 习题 10 的电路图

参考文献

[1] 熊光楞等. 连续系统仿真与离散事件系统仿真. 北京：清华大学出版社，1991
[2] 周炎勋等. 模拟与混合计算技术. 北京：国防工业出版社，1980
[3] 熊光楞等. 控制系统仿真与模型处理. 北京：科学出版社，1993
[4] 刘植桢等. 计算机控制. 北京：清华大学出版社，1981
[5] 绪方胜彦著. 卢伯英等译. 现代控制工程. 北京：科学出版社，1978
[6] 王红卫. 建模与仿真. 北京：科学出版社，2002
[7] Zeigler. Theory of modeling and simulation. John Wiley & Sons, 1976
[8] 瓦拉赫 J，辛格尔 K 著. 汪蕙等译，电路分析和设计的计算机方法. 北京：科学出版社，1992
[9] 陈立平等. 机械系统动力学分析及 ADAMS 应用教程. 北京：清华大学出版社，2005
[10] 肖田元. 虚拟制造. 北京：清华大学出版社，2004

第3章
经典的连续系统仿真建模方法学

本章讨论经典的连续系统数字仿真的原理与方法，内容包括连续系统离散化原理数字仿真的基本概念、经典的单步数值积分法、经典的线性多步法等。在数字计算机上进行连续系统仿真，首先要将连续模型离散化，因此3.1节讨论连续系统离散化原理，这是连续系统仿真的基础；然后，3.2节对经典的数值积分法——龙格-库塔法及其它典型的数值积分法仿真建模原理进行详细分析，并通过实例说明其应用要点；3.3节介绍经典的多步法；3.4节讨论基于数值积分原理的各种方法的稳定性；3.5节讨论病态系统仿真建模技术；3.6节讨论间断特性仿真建模。

3.1 连续系统离散化原理

在数字计算机上对连续系统进行仿真时，首先遇到的问题是，数字计算机的数值及时间均具有离散性，而被仿真系统的数值及时间却具有连续性，后者如何用前者来实现。

从根本意义上讲，数字计算机所进行的数值计算仅仅是“数字”计算，它表示数值的精度受限于字长，这将引入舍入误差；另一方面，这种计算是按指令一步一步进行的，因而，还必须将时间离散化，这样就只能得到离散时间点上的系统性能。用数字仿真的方法对微分方程的数值积分是通过某种数值计算方法来实现的。任何一种计算方法都只能是原积分的一种近似。因此，连续系统仿真在本质上是从时间、数值两个方面对原系统进行离散化，并选择合适的数值计算方法来近似积分运算，由此得到离散模型来近似原连续模型。如何保证离散模型的计算结果从原理上的确能代表原系统的行为，这是连续系统数字仿真首先必须解决的问题。

设系统模型(参见图3.1)为$\dot{y}=f(y,u,t)$，其中$u(t)$为输入变量，$y(t)$为系统变量，令仿真时间间隔为h，离散化后的输入变量为$\hat{u}(t_k)$，系统变量为$\hat{y}(t_k)$，其中t_k表示$t=kh$。如果$\hat{u}(t_k)\approx u(t_k)$，$\hat{y}(t_k)\approx y(t_k)$，即$e_u(t_k)=\hat{u}(t_k)-u(t_k)\approx 0$，$e_y(t_k)=\hat{y}(t_k)-y(t_k)\approx 0$(对所有

$k=0,1,2,\cdots$),则可认为两模型等价。

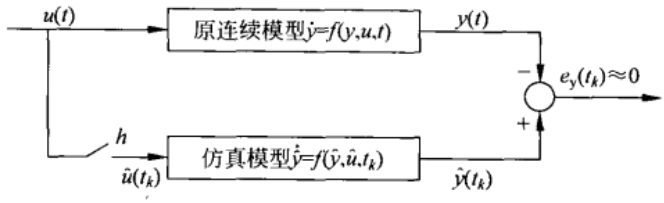


图 3.1 连续模型与离散模型等价示意

实际上,要完全保证 $e_u(t_k)=0, e_y(t_k)=0$ 是很困难的。进一步分析离散化引入的误差,随着计算机技术的发展,由计算机字长引入的舍入误差可以忽略,关键是数值积分算法,也称为仿真建模方法。相似原理用于仿真时,对仿真建模方法有三个基本要求:

(1) 稳定性 若原连续系统是稳定的,则离散化后得到的仿真模型也应是稳定的。关于稳定性的详细讨论将在 3.4 节中进行。

(2) 准确性 有不同的准确性评价准则,最基本的准则是

$$\text{绝对误差准则} \quad |e_y(t_k)| = |\hat{y}(t_k) - y(t_k)| \leqslant \delta$$

$$\text{相对误差准则} \quad |e_y(t_k)| = \left|\frac{\hat{y}(t_k) - y(t_k)}{\hat{y}(t_k)}\right| \leqslant \delta$$

其中 δ 表示规定的误差量。

(3) 快速性 如前所述,数字仿真是一步一步推进的,即由某一初始值 $y(t_0)$ 出发,逐步计算,得到 $y(t_1), y(t_2), \cdots, y(t_k)$,每一步计算所需时间决定了仿真速度。如果第 k 步计算对应的系统时间间隔为 $h_k=t_{k+1}-t_k$,计算机由 $y(t_k)$ 计算 $y(t_{k+1})$ 需要的时间为 T_k,那么,若 $T_k=h_k$ 称为实时仿真,则 $T_k<h_k$ 称为超实时仿真,而大多数情况是 $T_k>h_k$,对应于离线仿真。

连续系统数字仿真中最基本的算法是数值积分法。数值积分法是指对于形如 $\dot{y}=f(y,u,t)$ 的系统,已知系统变量 y 的初始条件 $y(t_0)=y_0$,现在要计算 y 随时间变化的过程 $y(t)$,可以在一系列离散点 $t_1, t_2, \cdots, t_N$ 上计算出未知函数 $y(t)$ 值 $y(t_1), y(t_2), \cdots, y(t_N)$ 的近似值 $y_1, y_2, \cdots, y_N$。

计算过程可以这样考虑(参见图 3.2):首先求出初始点 $y(t_0)=y_0$ 的 $f(t_0,y_0)$,对微分方程积分可以写作

$$y(t) = y_0 + \int_{t_0}^{t} f(t,y)\mathrm{d}t \tag{3.1}$$

图 3.2 所示曲线下的面积就是 $y(t)$,由于难以得到 $f(y,u,t)$ 积分的数值表达式,人们对数值积分法进行了长期探索,其中欧拉法是最经典的近似方法。

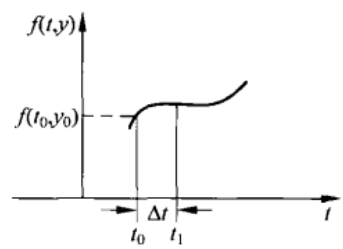


图 3.2 数值积分法原理

欧拉法用矩形面积近似表示积分结果,也就是当 $t=t_1$ 时,$y(t_1)$ 的近似值为 y_1

$$y(t_1) \approx y_1 = y_0 + \Delta t \cdot f(t_0, y_0) \tag{3.2}$$

重复上述做法,当 $t=t_2$ 时

$$y(t_2) \approx y_2 = y_1 + (t_2 - t_1) \cdot f(t_1, y_1)$$

所以,对任意时刻 t_{k+1},有

$$y(t_{k+1}) \approx y_{k+1} = y_k + (t_{k+1} - t_k) \cdot f(t_k, y_k) \tag{3.3}$$

令 $t_{k+1} - t_k = h_k$，称为第 k 步的计算步距。若积分过程中步距不变 $h_k = h$，称为定步长，如果变化则称为变步长。

为考察欧拉法的可用性，首先应该知道它所提供的数值解是否能足够近似真正的解，这就必须估计数值解与真实解之间的误差。欧拉法是将一个步长内的导函数取值作为起始点的值并保持不变，即用差商代替导函数。这种近似替代所产生的误差称为截断误差。可以证明，欧拉法的截断误差正比于 h^2。

为进一步提高计算精度，人们提出了“梯形法”。梯形法近似积分形式如式(3.4)所示，令：$t_{k+1} - t_k = h_k = h$，已知 $t = t_k$ 时 $y(t_k)$ 的近似值 y_k，那么

$$y(t_{k+1}) \approx y_{k+1} = y_k + \frac{1}{2}h[f(t_k, y_k) + f(t_{k+1}, y_{k+1})] \tag{3.4}$$

可见，梯形法是隐函数形式。采用这种积分方法，其最简单的预报—校正方法是用欧拉法估计初值，用梯形法校正，即

$$y_{k+1}^{(i+1)} \approx y_k + \frac{1}{2}h[f(t_k, y_k) + (t_{k+1}, y_{k+1}^{(i)})] \tag{3.5}$$

$$y_{k+1}^{(i)} \approx y_k + h \cdot f(t_k, y_k) \tag{3.6}$$

式(3.6)称作预报公式，式 (3.5)称为校正公式。用欧拉法估计 $y_{k+1}^{(i)}$ 的值，代入校正公式得到 y_{k+1} 的校正值 $y_{k+1}^{(i+1)}$。设 ε 是规定的足够小的正数，称作允许误差，若 $i=0$，$i+1=1$ 称作第一次校正；$i=1$，$i+1=2$ 称作第二次校正；通过反复迭代，直到满足 $|y_{k+1}^{i+1} - y_{k+1}^{i}| \leqslant \varepsilon$，这时 $y_{k+1}^{(i+1)}$ 是满足误差要求的校正值。

上述方法是针对式(3.3)所示的微分方程在已知初值情况下进行求解，因此也称为微分方程初值问题数值计算法，为统一起见，本书中称为数值积分法。

数值积分法采用递推方式进行运算，而采用不同的积分方法会引进不同的计算误差，为了提高计算精度，往往会增加运算量。就同一种积分算法而言，为提高计算精度，需减小积分步距 h，这样将降低计算速度。因此，计算精度和速度是连续系统仿真中常见的一对矛盾，也是数字仿真中要求解决的问题之一。也就是说，选择合适的算法、合适的软、硬件环境，在保证计算精度的前提下，考虑怎样提高仿真的速度。

经典的数值积分法可分为单步法与多步法，下面我们将分别介绍这两类方法中最常用的算法。

3.2 龙格-库塔法

3.2.1 龙格-库塔法基本原理

上一节已经讲过，在连续系统的仿真中，主要的数值计算工作是对形如 $\frac{dy}{dt} = f(t, y)$ 的一阶微分方程进行求解。因为

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(t, y)\,dt$$

若令 $y_k \approx y(t_k)$，$Q_k \approx \int_{t_k}^{t_{k+1}} f(t,y)\mathrm{d}t$，则有

$$y(t_{k+1}) \approx y_{k+1} = y_k + Q_k \tag{3.7}$$

因此主要问题就是如何对 Q_k 进行数值求解，即如何对 $f(t,y)$进行积分。这通常称作“右端函数”计算问题。已知 $y(t_0)=y_0$，假设我们从 t_0 跨出一步，$t_1=t_0+h$，t_1 时刻为 $y_1=y(t_0+h)$，可以在 t_0 附近展开泰勒级数，只保留 h^2 项，则有

$$y_1 = y_0 + f(t_0,y_0)h + \frac{1}{2}\left(\frac{\partial f}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}t} + \frac{\partial f}{\partial t}\right)\Bigg|_{t_0} h^2 \tag{3.8}$$

在式 (3.8)中括号后的下标表示括号中的函数将用 $t=t_0$，$y=y_0$ 取值。我们假设这个解可以写成如下形式

$$y_1 = y_0 + (a_1K_1 + a_2K_2)h \tag{3.9}$$

其中

$$K_1 = f(t_0,y_0)$$

$$K_2 = f(t_0+b_1h, y_0+b_2K_1h)$$

对 K_2 式右端的函数在 $t=t_0$，$y=y_0$ 处展成泰勒级数，保留 h 项，可得

$$K_2 \approx f(t_0,y_0) + \left(b_1\frac{\partial f}{\partial t} + b_2K_1\frac{\partial f}{\partial y}\right)\Bigg|_{t_0} h$$

将 K_1，K_2 代入式(3.9)，则有

$$y_1 = y_0 + a_1hf(t_0,y_0) + a_2h\left[f(t_0,y_0) + \left(b_1\frac{\partial f}{\partial t} + b_2K_1\frac{\partial f}{\partial y}\right)\Bigg|_{t_0} h\right]$$

将上式与式(3.8)进行比较，可得

$$a_1 + a_2 = 1, \quad a_2b_1 = 1/2, \quad a_2b_2 = 1/2$$

可见有四个未知数 a_1，a_2，b_1，b_2，但只有三个方程，因此有无穷多个解，若限定 $a_1=a_2$，则可得其中一个解

$$a_1 = a_2 = \frac{1}{2}, \quad b_1 = b_2 = 1$$

将它们代入式(3.9)可得一组计算公式

$$y_1 = y_0 + \frac{h}{2}(K_1 + K_2) \tag{3.10}$$

其中，$K_1=f(t_0,y_0)$，$K_2=f(t_0+h,y_0+K_1h)$。若写成一般递推形式，即

$$y(t_{k+1}) \approx y_{k+1} = y_k + \frac{h}{2}(K_1 + K_2) \tag{3.11}$$

其中，$K_1=f(t_k,y_k)$，$K_2=f(t_k+h,y_k+K_1h)$。

由于式(3.8)只取了 h，h^2 两项，而将 h^3 以上的高阶项略去了，所以这种递推公式的截断误差正比于 h 的三次方，又因计算时只取了 h 及 h^2 项，故这种方法被称为二阶龙格-库塔法(简称 RK-2)。

根据上述原理，若在展成泰勒级数时保留 h，h^2，h^3 及 h^4 项，那么可得截断误差正比于 h^5 的四阶龙格-库塔法(简称 RK-4)公式

$$y(t_{k+1}) \approx y_{k+1} = y_k + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4) \tag{3.12}$$

其中

$$K_1 = f(t_k, y_k)$$

$$K_2 = f\left(t_k + \frac{h}{2}, y_k + \frac{h}{2}K_1\right)$$

$$K_3 = f\left(t_k + \frac{h}{2}, y_k + \frac{h}{2}K_2\right)$$

$$K_4 = f(t_k + h, y_k + hK_3)$$

由于这组计算公式有较高的精度，所以在数字仿真中应用较为普遍。

为了帮助读者更好地掌握这种方法的使用，下面我们对龙格-库塔法的特点作一些介绍。

由于在解 a_1, a_2, b_1, b_2 时，可以得到许多种龙格-库塔公式，经常使用的除式(3.11)及式(3.12)给出的两组公式外，还有：

$$y_{k+1} = y_k + K_2 h \tag{3.13}$$

其中

$$K_1 = f(t_k, y_k)$$

$$K_2 = f\left(t_k + \frac{h}{2}, y_k + \frac{h}{2}K_1\right)$$

$$y_{k+1} = y_k + \frac{h}{8}(K_1 + 3K_2 + 3K_3 + K_4) \tag{3.14}$$

其中

$$K_1 = f(t_k, y_k)$$

$$K_2 = f\left(t_k + \frac{h}{3}, y_k + \frac{h}{3}K_1\right)$$

$$K_3 = f\left(t_k + \frac{2}{3}h, y_k - \frac{h}{3}K_1 + hK_2\right)$$

$$K_4 = f(t_k + h, y_k + K_1 h - K_2 h + K_3 h)$$

式(3.13)为二阶龙格-库塔公式，式(3.14)是 Shampine 提出的，称为 RKS-34 公式。

对上述公式可以这样解释：因为 $y(t)$ 满足 $\mathrm{d}y/\mathrm{d}t = f(t, y)$ 这个微分方程，已知 (t_k, y_k) 处 $y(t)$ 的导数为 $f(t_k, y_k) = K_1$。根据外推原理，假设在 $t_k \sim t_k + h$ 这个区间中 $y(t)$ 的导数不变，则可得 $y(t)$ 在 $t = t_k + h$ 的估计值 $y_k + K_i h$。令 K_i 为这个估计值处的导数，因此式(3.11)就是按 $(K_1 + K_2)/2$ 这个导数线性外推而得的 y_{k+1}。而式(3.12)则是 4 点导数的加权平均和，由于第 2 及第 3 点是以 $h/2$ 为步长，精度较高，所以加权平均时各取 2 份，第 1、4 点各取 1 份。不论几阶龙格-库塔法，它们的计算公式总是由两部分组成的，即是上一步的结果 y_k 及步长 h 乘以 $t_k - t_{k+1}$ 中各点导数的加权平均和。因此各种龙格-库塔法可以写成如下一般形式

$$y_{k+1} = y_k + h\sum_{i=1}^{s} C_i K_i \tag{3.15}$$

其中

$$K_i = f\left(t_k + a_i h, y_k + h\sum_{j=1}^{i-1} b_{ij} K_j\right) \qquad (i = 1,2,\cdots,s)$$

式中各系数满足以下关系

$$\begin{cases} a_1 = 0 \\ a_i = \sum\limits_{j=1}^{i-1} b_{ij} \quad (i = 2,3,\cdots,s) \\ \sum\limits_{i=1}^{s} C_i = 1 \end{cases} \tag{3.16}$$

一般来讲，为了减少计算量，总是希望减少每步计算右端函数 f 的次数（即减少计算 K 的次数）。可以证明，一阶龙格-库塔公式至少要计算一次右端函数，即式(3.11)中的 s（称为级数）至少等于 1，而二阶公式 $s_{\min}=2$；三阶公式 $s_{\min}=3$；四阶公式 $s_{\min}=4$；依此类推。有时为了某种特殊需要，可以选择 $s>s_{\min}$ 的计算公式。

表 3.1 为常见的各阶龙格-库塔公式，第三项是四阶 5 级 RK-4 计算公式（简称 RKM-45）。其中，后 3 项有误差估计公式，所以可以用误差来控制积分步长，即每积分一步，利用误差估计公式估计出这一步的截断误差。若它大于允许的误差，则缩小步长，反之可以放大步长。

表 3.1 常见的龙格-库塔公式

名　称	公　式	特　点
二阶龙格-库塔法	$y_{k+1}=y_k+\frac{h}{2}(K_1+K_2)$ $K_1=f(t_k,y_k)$ $K_2=f(t_k+h,y_k+hK_1)$	
四阶龙格-库塔法	$y_{k+1}=y_k+\frac{h}{6}(K_1+2K_2+2K_3+K_4)$ $K_1=f(t_k,y_k)$ $K_2=f\left(t_k+\frac{h}{2},y_k+\frac{h}{2}K_1\right)$ $K_3=f\left(t_k+\frac{h}{2},y_k+\frac{h}{2}K_2\right)$ $K_4=f(t_k+h,y_k+hK_3)$	
四阶龙格-库塔-默森法	$y_{k+1}=y_k+\frac{h}{6}(K_1+4K_4+K_5)$ $K_1=f(t_k,y_k)$ $K_2=f\left(t_k+\frac{h}{3},y_k+\frac{h}{3}K_1\right)$ $K_3=f\left(t_k+\frac{h}{3},y_k+\frac{h}{6}(K_1+K_2)\right)$ $K_4=f\left(t_k+\frac{h}{2},y_k+\frac{h}{8}(K_1+3K_3)\right)$ $K_5=f\left(t_k+h,y_k+\frac{h}{2}(K_1+4K_4-3K_3)\right)$	有误差估计公式 $E_k=\frac{h}{6}(2K_1-9K_3+8K_4-K_5)$

续表

名　称	公　式	特　点
二阶龙格-库塔-费尔别格法	$y_{k+1}=y_k+\frac{h}{512}(K_1+510K_2+K_3)$ $K_1=f(t_k,y_k)$ $K_2=f\left(t_k+\frac{h}{2},y_k+\frac{h}{2}K_1\right)$ $K_3=f\left(t_k+h,y_k+\frac{h}{256}(K_1+255K_2)\right)$	有误差估计公式 $E_k=\frac{h}{512}(K_1-K_3)$
四阶龙格-库塔-夏普法	$y_{k+1}=y_k+\frac{h}{8}(K_1+3K_2+3K_3+K_4)$ $K_1=f(t_k,y_k)$ $K_2=f\left(t_k+\frac{h}{3},y_k+\frac{h}{3}K_1\right)$ $K_3=f\left(t_k+\frac{2}{3}h,y_k+\frac{h}{3}(-K_1+3K_2)\right)$ $K_4=f(t_k+h,y_k+h(K_1-K_2+K_3))$ $K_5=f\left(t_k+h,y_k+\frac{h}{8}(K_1+3K_2+3K_3+K_4)\right)$	有误差估计公式 $E_k=\frac{h}{32}(-K_1+3K_2-3K_3-3K_4+4K_5)$

所有龙格-库塔公式都有以下特点：

(1) 在计算 y_{k+1} 时只用到 y_k，而不直接用 y_{k-1}，y_{k-2} 等项。换句话说，在后一步的计算中，仅仅利用前一步的计算结果，所以称为单步法。显然它不仅能使存储量减小，而且此法可以自启动，即已知初值后，不必用别的方法来帮助，就能由初值逐步计算得到后续各时间点上的仿真值。

(2) 步长 h 在整个计算中并不要求固定，可以根据精度要求改变，但是在一步中，为计算若干个系数 K_i(俗称龙格-库塔系数)，则必须用同一个步长 h。

(3) 龙格-库塔法的精度取决于步长 h 的大小及方法的阶次。许多计算实例表明：为达到相同的精度，四阶方法的 h 可以比二阶方法的 h 大 10 倍，而四阶方法的每步计算量仅比二阶方法大一倍，所以总的计算量仍比二阶方法小。正是由于上述原因，一般系统仿真常用四阶龙格-库塔公式。值得指出的是：高于四阶的方法由于每步计算量增加较多，而精度提高不快，因此使用得也比较少。

(4) 若在展开泰勒级数时，只取 h 这一项，而将 h^2 以及 h^2 以上的项都略去，则可得

$$y_{k+1}=y_k+hf(t_k,y_k)$$

这就是欧拉公式，因此欧拉公式也可看作是一阶龙格-库塔公式。它的截断误差正比于 h^2，是精度最低的一种数值积分公式。

读者也许会问：如果我们将步长取得十分小，那么欧拉公式也可以获得很高的计算精度吗？从理论上讲是这样，但是实际上由于计算机字长有限，在计算中存在着舍入误差，它与计算次数成正比。计算步长很小，则一个系统的过渡过程的计算就会增加许多，舍入误差就会十分明显地表现出来，这样就很难保证有较高的精度。

3.2.2　龙格-库塔法的误差估计及步长控制

一个高精度的仿真方法必须将步长的控制作为必要的手段。实现步长控制涉及局部误

差估计和步长控制策略两方面的问题。本节将对数值积分中单步法的误差估计和步长控制作一介绍。

龙格-库塔数值积分方法的“误差估计”和“步长控制”的基本想法如图 3.3 所示，也就是每积分一步都设法估计出本步的计算误差 ε_k，然后判断是否满足允许误差 E，据此选择相应的步长控制策略，调整步长，再作下一步积分运算。

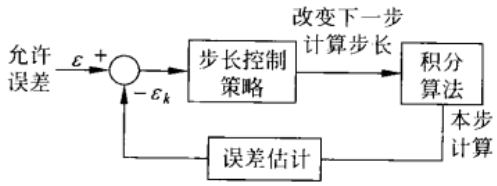


图 3.3 龙格-库塔法步长控制示意

1. 龙格-库塔法的误差估计

龙格-库塔法的误差估计，通常采用的方法是设法找到另一个低阶(一般是低一阶)的龙格-库塔公式，两个公式计算结果之差可以被看作是误差。例如，龙格-库塔-默森(Runge-Kutta-Merson)法计算公式为

$$y_{k+1} = y_k + \frac{h}{6}(K_1 + 4K_4 + K_5) \tag{3.17}$$

其中

$$K_1 = f(t_k, y_k)$$

$$K_2 = f\left(t_k + \frac{h}{3}, y_k + \frac{h}{3}K_1\right)$$

$$K_3 = f\left(t_k + \frac{h}{3}, y_k + \frac{h}{6}(K_1 + K_2)\right)$$

$$K_4 = f\left(t_k + \frac{h}{2}, y_k + \frac{h}{8}(K_1 + 3K_3)\right)$$

$$K_5 = f\left(t_k + h, y_k + \frac{h}{2}(K_1 + 4K_4 - 3K_3)\right)$$

它是一个四阶 5 级公式。另外还可推出一个三阶 4 级公式

$$\hat{y}_{k+1} = y_k + \frac{h}{6}(3K_1 - 9K_3 + 12K_4) \tag{3.18}$$

计算 K_1, K_3, K_4 的公式与式(3.17)相同。用 $\hat{y}_{k+1} - y_{k+1}$ 作为误差估计，即

$$E_k = \frac{h}{6}(2K_1 - 9K_3 + 8K_4 - K_5) \tag{3.19}$$

误差的精度为三阶。式(3.17)、式(3.18)及式(3.19)简称为 RKM-34 法。又比如，E. Fehlberg 推导出的五阶龙格-库塔-费尔别格法，它的计算公式为一个五阶 6 级方法，另外用了一个四阶 5 级方法求 $\hat{y}_{k+1}$，也是用 $\hat{y}_{k+1} - y_{k+1}$ 来估计误差，这一套计算公式被公认是对非病态系统进行仿真最为有效的方法之一。由于它是五阶精度、四阶的误差估计，因此被称为 RKF-45 阶公式对，简称 RK-45 法。具体表达式如下

$$\begin{cases} y_{k+1} = y_k + h\sum_{i=1}^{6} c_i K_i \\ K_i = f\left(t_k + a_i h, y_k + h\sum_{j=1}^{i-1} b_{ij} K_i\right) \end{cases} \quad (i = 1, 2, \cdots, 6) \tag{3.20}$$

式(3.20)中的各系数见表 3.2。

表 3.2 Fehlberg 公式中的各系数

i	a_i	b_{ij}					c_i	c_i^*
1	0						16/135	25/216
2	1/4	1/4					0	0
3	3/8	3/32	9/32				6 656/12 825	1 408/2 565
4	12/13	1 932/2 197	−7 200/2 197	7 296/2 197			28 561/56 430	2 197/4 104
5	1	439/216	−8	3 680/513	−845/4 104		−9/50	−1/5
6	1/2	−8/27	2	−3 544/2 566	1 859/4 104	−11/40	2/55	0

表中 c_i^* 表示四阶公式中的系数，因此误差估计为

$$E_k = \hat{y}_{k+1} - y_{k+1} = h\sum_{i=1}^{6}(c_i^* - c_i)K_i \tag{3.21}$$

当精度要求不太高时，RKF-12 法被广泛采用。它的计算公式为

$$y_{k+1} = y_k + \frac{h}{512}(K_1 + 510K_2 + K_3) \tag{3.22}$$

其中

$$\begin{cases} K_1 = f(t_k, y_k) \\ K_2 = f\left(t_k + \dfrac{h}{2}, y_k + \dfrac{h}{2}K_1\right) \\ K_3 = f\left(t_k + h, y_k + \dfrac{h}{256}(K_1 + 255K_2)\right) \end{cases}$$

用另一个一阶公式

$$\hat{y}_{k+1} = y_k + \frac{h}{256}(K_1 + 255K_2) \tag{3.23}$$

来估计误差

$$E_k = \hat{y}_{k+1} - y_{k+1} = \frac{h}{512}(K_1 - K_3) \tag{3.24}$$

RKM-34 法与 RKF-45 法是两种应用得极为广泛的数值积分算法。它们的共同特点是计算量较大，比如 RKM-34 法，每步要计算 5 次 f，而只获得四阶精度及三阶误差估计，RKF-45 法每步要计算 6 次 f，能获得五阶精度及四阶误差估计。对于一般仿真问题，中等精度即可满足要求，因此用 RKM-34 公式对就够了。1978 年，夏普勒(Shampine)提出了 RKS-34 公式对，它每次只计算 4 次 f 却能获得四阶精度及三阶误差估计，计算公式如下

$$y_{k+1} = y_k + \frac{h}{8}(K_1 + 3K_2 + 3K_3 + K_4) \tag{3.25}$$

其中

$$\begin{cases} K_1 = f(t_k, y_k) \\ K_2 = f\left(t_k + \dfrac{h}{3}, y_k + \dfrac{h}{3}K_1\right) \\ K_3 = f\left(t_k + \dfrac{2h}{3}, y_k + \dfrac{h}{3}(-K_1 + 3K_2)\right) \\ K_4 = f(t_k + h, y_k + h(K_1 - K_2 + K_3)) \end{cases}$$

另外引入了一个三阶公式

$$\hat{y}_{k+1} = y_k + \frac{h}{32}(3K_1 + 15K_2 + 9K_3 + K_4 + 4K_5) \tag{3.26}$$

其中

$$K_5 = f\left(t_k + h, y_k + \frac{h}{8}(K_1 + 3K_2 + 3K_3 + K_4)\right) \tag{3.27}$$

它正好是下一次计算 y_{k+1} 时的 K_1，因此只是在第一步要多算一次 f，以后仍每步计算四次 f。RKS-34 的误差估计为

$$E_k = \hat{y}_{k+1} - y_{k+1} = \frac{h}{32}(-K_1 + 3K_2 - 3K_3 - 3K_4 + 4K_5) \tag{3.28}$$

2. 步长控制

(1) 加倍-减半法

设定一个最小误差限和最大误差限，当估计的局部误差大于最大误差时将步长减半，并重新计算这一步；当误差在最大误差与最小误差之间时，步长不变；当误差小于最小误差时将步长加倍。

每步的局部误差通常取以下形式

$$e_k = E_k/(|\ y_k\ |+1) \tag{3.29}$$

其中 E_k 为利用本节的公式计算出来的误差估计。由式(3.29)可知，当 $|y_k|$ 较大时，e_k 是相对误差，而当 y_k 的绝对值很小时，e_k 就成了绝对误差。这样做的目的是避免当 y 的值很小时，e_k 变得过大。

上述步长控制的策略可以表示为下式

$$\text{如果}\begin{cases} e_k \geqslant \varepsilon_{\max} \\ \varepsilon_{\min} < e_k < \varepsilon_{\max} \\ \varepsilon_{\min} \geqslant e_k \end{cases} \qquad \text{则选择}\begin{cases} h_{k+1} = \dfrac{1}{2}h_k \\ h_{k+1} = h_k \\ h_{k+1} = 2h_k \end{cases} \tag{3.30}$$

其中 $\varepsilon_{\max}$ 和 $\varepsilon_{\min}$ 为最大、最小误差限。这种步长控制的方法简便易行，每步附加计算量小，但是这种方法不能实现每步都是最优步长。

(2) 最优步长法

为了使每个积分步在保证精度的前提下能取最大步长(或称最优步长)，可以设法根据本步误差的估计，近似确定下一步可能的最大步长。由于这种方法可以做到在规定的精度下取得最大步长，因此减少了计算量。具体策略如下：

给定相对误差限 ε_0，设本步步长为 h_k，本步相对误差估计值为 e_k(如式(3.29)所示)。假定所采用的积分算法为 K 阶，则式(3.29)中的 E_k 可表示为

$$E_k = \varphi(\zeta) \cdot h^K \tag{3.31}$$

其中 $\varphi(\zeta)$ 是 $f(t,y)$ 在积分区间 (t_k, t_k+h) 中某处变量偏导数的组合，通常可取 $\zeta = t_k$，因此有

$$e_k = \varphi(t_k) \cdot h^K/(|\ y_k\ |+1) \tag{3.32}$$

若 $e_k \leqslant \varepsilon_0$，则本步积分成功。现在来确定下一步的最大步长 h_{k+1}，假定 h_{k+1} 足够小，即可认为 $\varphi(t_k+h_{k+1}) \approx \varphi(t_k)$，故下一步的误差为

$$e_k = \varphi(t_{k+1}) \cdot h^K / (|\ y_{k+1}\ | + 1) \approx \varphi(t_k) \cdot h^K / (|\ y_k\ | + 1)$$

为使 $e_{k+1} \leqslant \varepsilon_0$，则有

$$h_{k+1} \approx (\varepsilon_0 (|\ y_k\ | + 1) / \varphi(t_k))^{1/K}$$

将式(3.32)代入上式，得

$$h_{k+1} \approx (\varepsilon_0 \cdot h^K / e_k)^{1/K} = (\varepsilon_0 / e_k)^{1/K} \cdot h \tag{3.33}$$

若 $e_k > \varepsilon_0$，则本步失败。此时仍可采用式(3.33)，但它表示重新积分的本步步长。由于我们假定了 h_{k+1} 足够小，因此 $\varphi(t_k)$ 基本不变，故必须限制步长的无限放大和缩小，一般可限制 h 的最大放、缩系数为 10，即要求

$$0.1 h_k < h_{k+1} < 10 \cdot h_k \tag{3.34}$$

当 f 函数中含有间断特性时，采用上述两种控制方法在间断点附近会出现步长频繁放大、缩小的振荡现象。由于最优步长控制方法是以本步误差外推下一步步长，因此振荡现象更为严重。解决这一问题的方法将在 3.6 节讨论。

3.2.3 实时龙格-库塔法

仿真模型的运行速度往往与实际系统运行的速度不同。然而，当有实际的装置或被训练的人介入仿真过程时，就往往要求仿真模型的运行速度与实际系统运行的速度保持一致，这称为实时仿真。

一般的数值积分法难以满足实时仿真的要求，这不仅仅是因为由这些方法所得到的模型的执行速度较慢，而且这些方法的机理也不符合实时仿真的特点。下面让我们以二阶龙格-库塔法为例来分析。假设对如下一般形式的系统进行仿真

$$\frac{dy}{dt} = f(y, u(t)) \tag{3.35}$$

RK-2 公式如下

$$y_{k+1} = y_k + \frac{h}{2}(K_1 + K_2) \tag{3.36}$$

$$K_1 = f(t_k, u_k, y_k)$$

$$K_2 = f(t_{k+1}, u_{k+1}, y_k + K_1 h)$$

即在一个计算步内分两子步：首先在 t_k 时刻利用当前的 u_k，y_k 计算 K_1。假定在 $h/2$ 的时间内计算机正好计算一次右端函数 f，然后，在 $t_k + h/2$ 时刻计算 K_2，尽管此时 $y_{k+1/2}$ 已经得到，但 u_{k+1} 无法得到。实时仿真除了要满足执行速度的要求外，还要求实时接收外部输入，并实时产生输出。为此，或对 u_{k+1} 也进行预报(这将加大仿真误差)，或将仿真执行延迟 $h/2$。后者的 RK-2 公式的计算流程可表示成图 3.4。这就是说，输出要滞后半个计算步距。

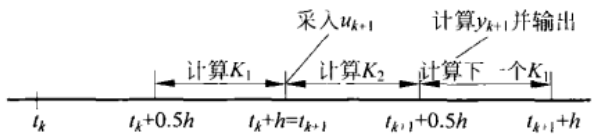


图 3.4 RK-2 的计算流程

与此相类似，式(3.12)所示的 RK-4 公式也不适合于实时仿真，读者可自行分析。

为了克服这个缺陷，人们提出了如下形式的实时二阶龙格-库塔法

$$
\begin{aligned}
y_{k+1} &= y_k + hK_2 \\
K_1 &= f(t_k, u_k, y_k) \\
K_2 &= f\left(t_{k+1/2}, u_{k+1/2}, y_k + \frac{h}{2}K_1\right)
\end{aligned}
\tag{3.37}
$$

它的计算流程如图 3.5 所示。由图不难看出，首先在 t_k 时刻利用当前的 u_k, y_k 计算 K_1。然后，在 $t_k + h/2$ 时刻计算 K_2，此时 $y_{k+1/2}$ 已经得到，由于计算一次右端函数 f 需要 $h/2$ 的时间，$u_{k+1/2}$ 也可得到，K_2 的计算就不会引入新的误差，并能实时输出 y_{k+1}。

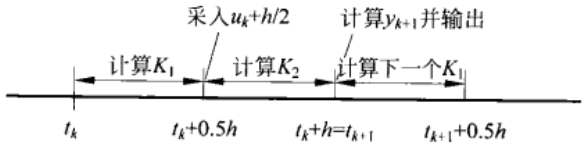


图 3.5 实时 RK-2 公式计算流程

此外，还有几种高阶的实时 RK 公式，例如，四阶五级的实时计算公式如下

$$
y_{k+1} = y_k + \frac{h}{24}[-K_1 + 15K_2 - 5K_3 + 5K_4 + 10K_5] \tag{3.38}
$$

其中

$$
\begin{aligned}
K_1 &= f(t_k, y_k) \\
K_2 &= f\left(t_k + \frac{h}{5}, y_k + \frac{h}{5}K_1\right) \\
K_3 &= f\left(t_k + \frac{2h}{5}, y_k + \frac{2h}{5}K_1\right) \\
K_4 &= f\left(t_k + \frac{3h}{5}, y_k - \frac{2h}{5}K_1 + hK_2\right) \\
K_5 &= f\left(t_k + \frac{4h}{5}, y_k + \frac{3h}{10}K_1 + \frac{h}{2}K_4\right)
\end{aligned}
$$

该公式的特点是：将一个仿真步分为 5 个子步，每个子步均能对外部输入实时采样，并保证了实时输出。它的计算流程如图 3.6 所示。

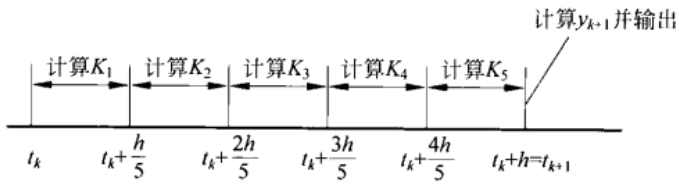


图 3.6 实时 RK-45 公式计算流程

3.3 线性多步法

龙格-库塔法也称作“单步法”，原因是在计算 y_{k+1} 时只用到 y_k，而不直接用 y_{k-1}, y_{k-2} 等项。也就是说，在后一步的计算中，仅仅利用前一步的计算结果。从信息利用的观点来看，

单步法对信息的利用率是不高的。“线性多步法”则是利用多步信息来计算下一步的值。

3.3.1 线性多步法基本原理

所有线性多步法的基本原理都是利用一个多项式去匹配变量的若干已知值和它们的导数值。本节介绍实现这种匹配的统一理论。在此均假定步长 h 为常数。

设 $t_n, t_{n+1}, \cdots, t_{n+k-1}$ 时刻的 $y_n, y_{n+1}, \cdots, y_{n+k-1}$ 和 $\dot{y}_n, \dot{y}_{n+1}, \cdots, \dot{y}_{n+k-1}$ 已知，而 y_{n+k}，$\dot{y}_{n+k}$ 未知，若由 $y_n, y_{n+1}, \cdots, y_{n+k-1}$ 和 $\dot{y}_n, \dot{y}_{n+1}, \cdots, \dot{y}_{n+k-1}$ 来计算 $y_{n+k}, \dot{y}_{n+k}$，则称为预报；若 y_{n+k} 也已知，由它们来计算 $\dot{y}_{n+k}$，则称为校正。

设拟采用的多项式具有以下形式

$$\begin{cases} y_m(t) = \sum_{i=0}^{m} d_i \left(\frac{t_{n+k}-t}{h}\right)^i = \sum_{i=0}^{m} d_i \tau^i \\ \dot{y}_m(t) = -\frac{1}{h}\sum_{i=1}^{m} i d_i \left(\frac{t_{n+k}-t}{h}\right)^{i-1} = -\frac{1}{h}\sum_{i=1}^{m} i d_i \tau^{i-1} \end{cases} \tag{3.39}$$

其中 d_i 是待定系数，$\tau=(t_{n+k}-t)/h$。显然，在 t_{n+k} 时刻，$\tau=0$，可得到

$$y_{n+k} = d_0 \tag{3.40}$$

$$\dot{y}_{n+k} = -\frac{1}{h} d_1 \tag{3.41}$$

为由 $y_n, y_{n+1}, \cdots, y_{n+k-1}$ 和 $\dot{y}_n, \dot{y}_{n+1}, \cdots, \dot{y}_{n+k-1}$ 确定 $d_i (i=0,1,2,\cdots,m)$，需要 $m+1$ 个独立方程。该 $m+1$ 个方程可根据以下等式导出

$$\begin{cases} y_m(t) = y_{n+k-j} = \sum_{i=0}^{m} d_i \left(\frac{t_{n+k}-t_{n+k-j}}{h}\right)^i = \sum_{i=0}^{m} d_i j^i \\ \dot{y}_m(t) = \dot{y}_{n+k-j} = -\frac{1}{h}\sum_{i=1}^{m} i d_i \left(\frac{t_{n+k}-t_{n+k-j}}{h}\right)^{i-1} = -\frac{1}{h}\sum_{i=1}^{m} i d_i j^{i-1} \end{cases} \quad (j=1,2,\cdots) \tag{3.42}$$

1. 预报公式

由 $y_n, y_{n+1}, \cdots, y_{n+k-1}$ 和 $\dot{y}_n, \dot{y}_{n+1}, \cdots, \dot{y}_{n+k-1}$ 来预报 $y_{n+k}, \dot{y}_{n+k}$，可令 $m=2k-1$，从式(3.42)得到如下方程组

$$\begin{aligned} & d_0^p + d_1^p + d_2^p + \cdots + d_m^p = y_{n+k-1} \\ & d_0^p + 2d_1^p + 4d_2^p + \cdots + 2^m d_m^p = y_{n+k-2} \\ & d_0^p + 3d_1^p + 3^2 d_2^p + \cdots + 3^m d_m^p = y_{n+k-3} \\ & \vdots \\ & d_1^p + 2d_2^p + \cdots + m d_m^p = -h\dot{y}_{n+k-1} \\ & d_1^p + 2\times 2d_2^p + \cdots + m\times 2^{m-1} d_m^p = -h\dot{y}_{n+k-2} \\ & d_1^p + 2\times 3d_2^p + \cdots + m\times 3^{m-1} d_m^p = -h\dot{y}_{n+k-3} \\ & \vdots \end{aligned}$$

将其写成矩阵形式

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 2^2 & 2^3 & \cdots & 2^m \\ 1 & 3 & 3^2 & 3^3 & \cdots & 3^m \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & k & k^2 & k^3 & \cdots & k^m \\ 0 & 1 & 2 & 3 & \cdots & m \\ 0 & 1 & 2\times 2 & 3\times 2^2 & \cdots & m\times 2^{m-1} \\ 0 & 1 & 2\times 3 & 3\times 3^2 & \cdots & m\times 3^{m-1} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 1 & 2\times k & 3\times k^2 & \cdots & m\times k^{m-1} \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{k-1} \\ d_k \\ d_{k+1} \\ d_{k+2} \\ \vdots \\ d_m \end{bmatrix} = \begin{bmatrix} y_{n+k-1} \\ y_{n+k-2} \\ y_{n+k-3} \\ \vdots \\ y_n \\ -h\dot{y}_{n+k-1} \\ -h\dot{y}_{n+k-2} \\ -h\dot{y}_{n+k-3} \\ \vdots \\ -h\dot{y}_n \end{bmatrix} \tag{3.43}$$

简记为

$$\boldsymbol{V}^p\boldsymbol{d}^p = \boldsymbol{Z}^p \tag{3.44}$$

其中上标 p 表示预报。形式上，其解为

$$\boldsymbol{d}^p = (\boldsymbol{V}^p)^{-1}\boldsymbol{Z}^p \tag{3.45}$$

由于 $\boldsymbol{V}^p$ 为常数阵，其逆存在，$\boldsymbol{Z}$ 向量中的各元素为已知值，因而 $\boldsymbol{d}$ 向量的各元素值可计算得到，从而由 $y_{n+k}=d_0$，$\dot{y}_{n+k}=-\frac{1}{h}d_1$ 得到下一时刻的预报值。

然而，利用式(3.45)来预报有两个缺点：首先，每次计算得到的 $\boldsymbol{d}$ 向量各元素只有 d_0，d_1 是所需要的，其他元素的计算成为多余；第二，每一步 $\boldsymbol{Z}$ 的值均发生变化，得不到 y_{n+k}，$\dot{y}_{n+k}$ 与 $y_n, y_{n+1}, \cdots, y_{n+k-1}$ 和 $\dot{y}_n, \dot{y}_{n+1}, \cdots, \dot{y}_{n+k-1}$ 的显式表达式。为此，我们对式(3.44)进行适当变换。

定义：$\boldsymbol{e}_0^{\mathrm{T}}=(1,0,\cdots,0)$ 为 $(m+1)\times 1$ 的列向量，上标 T 表示转置。将 $\boldsymbol{e}_0^{\mathrm{T}}$ 左乘式(3.45)可得

$$y_{n+k} = \boldsymbol{e}_0^{\mathrm{T}}\boldsymbol{d}^p = \boldsymbol{e}_0^{\mathrm{T}}(\boldsymbol{V}^p)^{-1}\boldsymbol{Z}^p \tag{3.46}$$

再定义辅助变量 $\boldsymbol{\phi}^p$

$$(\boldsymbol{\phi}^p)^{\mathrm{T}} = \boldsymbol{e}_0^{\mathrm{T}}(\boldsymbol{V}^p)^{-1} \tag{3.47}$$

此式可改写为

$$(\boldsymbol{V}^p)^{\mathrm{T}}\boldsymbol{\phi}^p = \boldsymbol{e}_0 \tag{3.48}$$

向量 $\boldsymbol{\phi}^p$ 的元素可划分为两个组

$$\boldsymbol{\phi}^p = [a_1^p, a_2^p, \cdots, a_k^p, \vdots\ b_1^p, b_2^p, \cdots, b_k^p]^{\mathrm{T}} \tag{3.49}$$

例如，设 $k=3$，则式(3.48)为

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 3 & 1 & 1 & 1 \\ 1 & 2^2 & 3^2 & 2 & 2\times 2 & 2\times 3 \\ 1 & 2^3 & 3^3 & 3 & 3\times 2^2 & 3\times 3^2 \\ 1 & 2^4 & 3^4 & 4 & 4\times 2^3 & 4\times 3^3 \\ 1 & 2^5 & 3^5 & 5 & 5\times 2^4 & 5\times 3^4 \end{bmatrix} \begin{bmatrix} a_1^p \\ a_2^p \\ a_3^p \\ b_1^p \\ b_2^p \\ b_3^p \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

可计算得到 $\boldsymbol{\phi}^p=[a_1^p, a_2^p, a_3^p, \vdots\ b_1^p, b_2^p, b_3^p]^{\mathrm{T}}=[-18, 9, 10, \vdots\ -9, -18, -3]^{\mathrm{T}}$。

值得注意的是，$\boldsymbol{\phi}^p$ 只依赖于 k，即本步前 y_{n+k-j} 和 $\dot{y}_{n+k-j}$ 的个数，而与它们的数值无关。这样，可以通过求解式(3.48)并代入式(3.46)得

$$y_{n+k} = \boldsymbol{e}_0^{\mathrm{T}}\boldsymbol{d}^p = (\boldsymbol{\phi}^p)^{\mathrm{T}}\boldsymbol{Z}^p \tag{3.50}$$

从而得到 y_{n+k} 的显式表达式

$$y_{n+k} = \sum_{j=1}^{k} a_j^p y_{n+k-j} - h\sum_{j=1}^{k} b_j^p \dot{y}_{n+k-j} \tag{3.51}$$

例 3.1　试推导用 y_{n+k-1}，$\dot{y}_{n+k-1}$，$\dot{y}_{n+k-2}$ 预报 y_{n+k} 的公式。

由于给出三个条件 y_{n+k-1}，$\dot{y}_{n+k-1}$，$\dot{y}_{n+k-2}$，故要用二阶多项式来近似，且 $(\boldsymbol{\phi}^p)^{\mathrm{T}}=(a_1^p, b_1^p, b_2^p)$。由 $(\boldsymbol{V}^p)^{\mathrm{T}}\boldsymbol{\phi}^p=\boldsymbol{e}_0$ 可列出

$$\boldsymbol{V}^p = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 1 & 4 \end{bmatrix}, \quad (\boldsymbol{V}^p)^{\mathrm{T}}\boldsymbol{\phi}^p = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix}\begin{bmatrix} a_1^p \\ b_1^p \\ b_2^p \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

不难计算出：$a_1^p=1$，$b_1^p=-3/2$，$b_2^p=1/2$，从而有

$$y_{n+k} = y_{n+k-1} + \frac{1}{2}h(3\,\dot{y}_{n+k-1} - \dot{y}_{n+k-2})$$

■

由该例可以看到，多步法公式是十分灵活的，它并不要求变量值的个数与其导数值的个数相同，因而可得到不同形式的多步法公式。

2. 校正公式

前面讨论了多步法预报公式，这是显式公式，它未包括 $\dot{y}_{n+k}$。在高精度仿真时，对该预报值应进行校正，即先预报得到 $\tilde{y}_{n+k}$，然后再用此值推出 $\dot{y}_{n+k}$。下面讨论校正公式。

由 $y_n, y_{n+1}, \cdots, y_{n+k-1}$ 和 y_{n+k}，以及 $\dot{y}_n, \dot{y}_{n+1}, \cdots, \dot{y}_{n+k-1}$ 来预报 $\dot{y}_{n+k}$，可令 $m=2k-1$，从式(3.42)得到如下方程组

$$\begin{aligned}
&d_0^c = y_{n+k}\\
&d_0^c + d_1^c + d_2^c + \cdots + d_m^c = y_{n+k-1}\\
&d_0^c + 2d_1^c + 4d_2^c + \cdots + 2^m d_m^c = y_{n+k-2}\\
&d_0^c + 3d_1^c + 3^2 d_2^c + \cdots + 3^m d_m^c = y_{n+k-3}\\
&\qquad\vdots\\
&d_1^c + 2d_2^c + \cdots + md_m^c = -h\,\dot{y}_{n+k-1}\\
&d_1^c + 2\times 2d_2^c + \cdots + m\times 2^{m-1} d_m^c = -h\,\dot{y}_{n+k-2}\\
&d_1^c + 2\times 3d_2^c + \cdots + m\times 3^{m-1} d_m^c = -h\,\dot{y}_{n+k-3}\\
&\qquad\vdots
\end{aligned}$$

其中上标 c 表示校正。将上述方程组写成矩阵形式

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 2^2 & 2^3 & \cdots & 2^m \\ 1 & 3 & 3^2 & 3^3 & \cdots & 3^m \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & k & k^2 & k^3 & \cdots & k^m \\ 0 & 1 & 2 & 3 & \cdots & m \\ 0 & 1 & 2\times 2 & 3\times 2^2 & \cdots & m\times 2^{m-1} \\ 0 & 1 & 2\times 3 & 3\times 3^2 & \cdots & m\times 3^{m-1} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_k \\ d_{k+1} \\ d_{k+2} \\ \vdots \\ d_m \end{bmatrix} = \begin{bmatrix} y_{n+k} \\ y_{n+k-1} \\ y_{n+k-2} \\ y_{n+k-3} \\ \vdots \\ y_n \\ -h\dot{y}_{n+k-1} \\ -h\dot{y}_{n+k-2} \\ -h\dot{y}_{n+k-3} \\ \vdots \end{bmatrix} \tag{3.52}$$

简记为

$$\boldsymbol{V}^c\boldsymbol{d}^c = \boldsymbol{Z}^c \tag{3.53}$$

由此可得

$$\boldsymbol{d}^c = (\boldsymbol{V}^c)^{-1}\boldsymbol{Z}^c \tag{3.54}$$

定义 $\boldsymbol{e}_1^{\mathrm{T}} = (0,1,0,\cdots,0)$ 为 $(m+1)\times 1$ 的列向量，上标 T 表示转置。将 $\boldsymbol{e}_1^{\mathrm{T}}$ 左乘式(3.54)可得

$$-h\dot{y}_{n+k} = \boldsymbol{e}_1^{\mathrm{T}}\boldsymbol{d}^c = \boldsymbol{e}_1^{\mathrm{T}}(\boldsymbol{V}^c)^{-1}\boldsymbol{Z}^c \tag{3.55}$$

再定义辅助变量 $\boldsymbol{\phi}^c$

$$(\boldsymbol{\phi}^c)^{\mathrm{T}} = \boldsymbol{e}_1^{\mathrm{T}}(\boldsymbol{V}^c)^{-1} \tag{3.56}$$

此式可改写为

$$(\boldsymbol{V}^c)^{\mathrm{T}}\boldsymbol{\phi}^c = \boldsymbol{e}_1 \tag{3.57}$$

向量 $\boldsymbol{\phi}^c$ 的元素可划分为两个组

$$\boldsymbol{\phi}^c = [a_0^c, a_1^c, \cdots, a_k^c, \vdots\ b_1^c, b_2^c, \cdots, b_k^c]^{\mathrm{T}} \tag{3.58}$$

同样，$\boldsymbol{\phi}^c$ 只依赖于 k，即先前 y_{n+k-j} 和 $\dot{y}_{n+k-j}$ 的个数，而与它们的数值无关。这样

$$-h\dot{y}_{n+k} = \boldsymbol{e}_1^{\mathrm{T}}\boldsymbol{d}^c = (\boldsymbol{\phi}^c)^{\mathrm{T}}\boldsymbol{Z}^c \tag{3.59}$$

从而得到

$$\dot{y}_{n+k} = -\frac{1}{h}\left(\sum_{j=0}^{k} a_j^c y_{n+k-j} - h\sum_{j=1}^{k} b_j^c \dot{y}_{n+k-j}\right) \tag{3.60}$$

例 3.2 用已知值 $y_{n+k-1}, y_{n+k-2}, y_{x+k-3}$ 预估 y_{n+k}，然后用 $y_{n+k}, y_{n+k-1}, y_{n+k-2}$ 推导 $\dot{y}_{n+k}$ 的校正公式。

预估公式的矩阵表达式为

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \begin{bmatrix} d_0^p \\ d_1^p \\ d_2^p \end{bmatrix} = \begin{bmatrix} y_{n+k-1} \\ y_{n+k-2} \\ y_{n+k-3} \end{bmatrix}$$

不难得到

$$(\boldsymbol{\phi}^p)^{\mathrm{T}} = [a_1^p, a_2^p, a_3^p] = [3, -3, 1]$$

则预报公式为

$$y_{n+k} = 3y_{n+k-1} - 3y_{n+k-2} + y_{n+k-3}$$

对校正公式，矩阵方程变为

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \begin{bmatrix} d_0^c \\ d_1^c \\ d_2^c \end{bmatrix} = \begin{bmatrix} y_{n+k} \\ y_{n+k-1} \\ y_{n+k-2} \end{bmatrix}$$

也不难得到

$$(\boldsymbol{\phi}^c)^{\mathrm{T}} = [a_0^c, a_1^c, a_2^c] = \left[-\frac{3}{2}, 2, -\frac{1}{2}\right]$$

则校正公式为

$$\dot{y}_{n+k} = -\frac{1}{h}\left(-\frac{3}{2}y_{n+k} + 2y_{n+k-1} - \frac{1}{2}y_{n+k-2}\right)$$

■

3.3.2　线性多步法误差分析

上一节我们推导了线性多步法的两种公式，并说明了每一种公式$\boldsymbol{\phi}$阵的取值仅仅依赖于变量及其导数的项数。本节我们进一步分析线性多步法的计算精度。

为了便于分析，对预报公式和校正公式，定义一个统一的表达式

$$\sum_{i=0}^{k} \alpha_i y_{n+k-i} - h\sum_{i=0}^{k} \beta_i \dot{y}_{n+k-i} = 0 \tag{3.61}$$

式中的某些系数可能为0。例如，当$\alpha_0 = 0$时显式预报$\dot{y}_{n+k}$，$\beta_0 = 0$时显式预报y_{n+k}，而$\beta_1 = \beta_2 = \cdots = \beta_k = 0$时称为后向差分公式(BDF)，而$\alpha_0$，$\beta_0$同时均不等于0时为隐式校正公式，$k$称为公式的阶次。例如典型的阿达姆氏(Adams)公式如下

$$y_{n+k} = y_{n+k-1} + h\sum_{j=0}^{k} b_j \dot{y}_{n+k-j}$$

由此可得$k=1,2,3,4$等各种情况下的显式及隐式阿达姆氏公式系数表，其中表3.3为显式，表3.4为隐式。

表3.3　显式阿达姆氏法系数表

k	b_0	b_1	b_2	b_3	b_4	截断误差
1	0	1	0	0	0	$\frac{1}{2}h^2\ddot{y}(\theta)$
2	0	3/2	−1/2	0	0	$\frac{5}{12}h^3\dddot{y}(\theta)$
3	0	23/12	−16/12	5/12	0	$\frac{9}{24}h^4y^{(4)}(\theta)$
4	0	55/24	−59/24	37/24	−9/24	$\frac{251}{720}h^5y^{(5)}(\theta)$

表 3.4 隐式阿达姆氏法系数表

k	b_0	b_1	b_2	b_3	b_4	截断误差
1	1	0	0	0	0	$-\frac{1}{2}h^2\ddot{y}(\theta)$
2	1/2	1/2	0	0	0	$-\frac{1}{12}h^3\dddot{y}(\theta)$
3	5/12	8/12	−1/12	0	0	$-\frac{1}{24}h^4y^{(4)}(\theta)$
4	9/24	19/24	−5/24	1/24	0	$-\frac{19}{720}h^5y^{(5)}(\theta)$

另一类多步法公式是后向差分(BDF)公式,其系数表如表 3.5、表 3.6 所示。其中表 3.5 为预报公式,表 3.6 为校正公式。

表 3.5 BDF 预估公式系数表 $y_{n+k}=\sum_{j=1}^{k+1}\alpha_j^P y_{n+k-j}$

k	α_1^P	α_2^P	α_3^P	α_4^P	α_5^P	α_6^P	α_7^P
1	2	−1	—	—	—	—	—
2	3	−3	1	—	—	—	—
3	4	−6	4	−1	—	—	—
4	5	−10	10	−5	1	—	—
5	6	−15	20	−15	6	−1	—
6	7	−21	35	−35	21	−7	1

表 3.6 BDF 校正公式系数表 $\dot{y}_{n+k}=-\frac{1}{h}\sum_{j=0}^{k}\alpha_j^C y_{n+k-j}$

k	α_0^C	α_1^C	α_2^C	α_3^C	α_4^C	α_5^C	α_6^C
1	−1	1	—	—	—	—	—
2	−3/2	2	−1/2	—	—	—	—
3	−11/6	3	−3/2	1/3	—	—	—
4	−25/12	4	−3	4/3	−1/4	—	—
5	−137/60	5	−5	10/3	−5/4	1/5	—
6	−147/60	6	−15/2	20/3	−15/4	6/5	−1/6

为了分析多步法公式的计算精度,假设变量各时间的精确值已经得到,将其代入式(3.61),可得

$$\sum_{i=0}^{k}\alpha_i y(t_{n+k}-ih)-h\sum_{i=0}^{k}\beta_i\dot{y}(t_{n+k}-ih)=0 \tag{3.62}$$

在 t_{n+k} 附近,将每个函数展开成泰勒级数

$$\begin{aligned}y(t_{n+k}-ih)&=y(t_{n+k})-\frac{ih}{1!}\dot{y}(t_{n+k})+\frac{(ih)^2}{2!}\ddot{y}(t_{n+k})+\cdots\\ \dot{y}(t_{n+k}-ih)&=\dot{y}(t_{n+k})-\frac{ih}{1!}\ddot{y}(t_{n+k})+\frac{(ih)^2}{2!}\dddot{y}(t_{n+k})+\cdots\end{aligned} \tag{3.63}$$

对所有 i,将式(3.63)代入式(3.62),合并同类项,可得

$$C_0 y(t_{n+k}) + hC_1 \dot{y}(t_{n+k}) + h^2 C_2 \ddot{y}(t_{n+k}) + \cdots + h^m C_m y^{(m)}(t_{n+k}) + \cdots \tag{3.64}$$

其中

$$C_0 = \alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_k$$

$$C_1 = (-1)^1 \left\{ \frac{1}{1!}(\alpha_1 + 2\alpha_2 + \cdots + k\alpha_k) + \frac{1}{0!}(\beta_0 + \beta_1 + \cdots + \beta_k) \right\}$$

$$C_2 = (-1)^2 \left\{ \frac{1}{2!}(\alpha_1 + 2^2\alpha_2 + \cdots + k^2\alpha_k) + \frac{1}{1!}(\beta_1 + 2\beta_2 + \cdots + k\beta_k) \right\}$$

$$C_3 = (-1)^3 \left\{ \frac{1}{3!}(\alpha_1 + 2^3\alpha_2 + \cdots + k^3\alpha_k) + \frac{1}{2!}(\beta_1 + 2^2\beta_2 + \cdots + k^2\beta_k) \right\}$$

$$\vdots$$

$$C_m = (-1)^m \left\{ \frac{1}{m!}(\alpha_1 + 2^m\alpha_2 + \cdots + k^m\alpha_k) + \frac{1}{(m-1)!}(\beta_1 + 2^{m-1}\beta_2 + \cdots + k^{m-1}\beta_k) \right\}$$

对 p 阶公式，要求 $C_0, C_1, C_2, \cdots, C_p$ 均为0，则

$$\sum_{i=0}^{k} \alpha_i y(t_{n+k} - ih) - h \sum_{i=0}^{k} \beta_i \dot{y}(t_{n+k} - ih) = h^{p+1} C_{p+1} y^{(p+1)}(t_{n+k}) + \cdots \tag{3.65}$$

下面，如果我们能证明上一节推导出的公式若能满足求 $C_0, C_1, C_2, \cdots, C_p$ 均为0的条件，就能得出这些公式的截断误差满足式(3.65)。以三阶公式为例，将前面所得到的 $C_0, C_1, C_2, \cdots, C_p$ 表达式为0，并写成矩阵形式

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2^2 & 3^2 & 0 & 2\times 1 & 2\times 2 & 2\times 3 \\ 0 & 1 & 2^3 & 3^3 & 0 & 3\times 1^2 & 3\times 2^2 & 3\times 3^2 \\ 0 & 1 & 2^4 & 3^4 & 0 & 4\times 1^3 & 4\times 2^3 & 4\times 3^3 \\ 0 & 1 & 2^5 & 3^5 & 0 & 5\times 1^4 & 5\times 2^4 & 5\times 3^4 \\ 0 & 1 & 2^6 & 3^6 & 0 & 6\times 1^5 & 6\times 2^5 & 6\times 3^5 \\ 0 & 1 & 2^7 & 3^7 & 0 & 7\times 1^6 & 7\times 2^6 & 7\times 3^6 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

先讨论预报公式，由于 $a_0 = -\alpha_0 = 1$ 和 $b_0 = \beta_0 = 0$，这意味着要将上述矩阵的第1列移到等式的右边，并去掉第5列。为使矩阵成为方阵，将其最后两行也去掉，其结果与用于推导预报公式的矩阵方程完全一样。

对校正公式，可采用类似的办法，只是 $b_0 = -\beta_0 = 1$，这样要将第5列移到等式的右边；若假定 $\alpha_3 = 0$，则可去掉第4列。同样为得到方阵，去掉最后两行，得到的就是三阶校正公式。

这就表明，上一节导出的预报与校正公式的截断误差系数可以用式(3.65)来计算，即

$$C_{p+1} = (-1)^{p+1} \left\{ \frac{1}{(p+1)!}(\alpha_1 + 2^{p+1}\alpha_2 + \cdots + k^{p+1}\alpha_k) + \frac{1}{p!}(\beta_1 + 2^p\beta_2 + \cdots + k^p\beta_k) \right\} \tag{3.66}$$

3.3.3 关于变步长的讨论

由于多步法用到了当前步之前若干步的变量及(或)其导函数的值，如果下一步要改变

步长，则必须按照新的步长重新启动。下面我们以 BDF 公式为例讨论多步法变步长问题。

设已知 k 个在 $t_n, t_{n+1}, \cdots, t_{n+k-1}$ 时刻的函数值 $y_n, y_{n+1}, \cdots, y_{n+k-1}$。预估公式使用这些值，为$(k-1)$阶。一旦获得预估值 y_{n+k}，则校正公式就使用这个新值，同时去掉 y_n。设当前的步长为 $h_k = t_{n+k} - t_{n+k-1}$。通过这些已知点的拟合多项式为

$$y_{k-1}(t) = \sum_{i=0}^{k-1} d_i \left(\frac{t_{n+k} - t}{h_k} \right)^i = \sum_{i=0}^{k-1} d_i \tau^i \tag{3.67}$$

对于各时刻点 t_{n+k-j} 的 y_{n+k-j}，引入符号

$$\tau_j = \frac{t_{n+k} - t_{n+k-j}}{h_k}$$

因而

$$y_{n+k-j} = \sum_{i=0}^{k-1} d_i \tau_j^i \tag{3.68}$$

因为对于 $t = t_{n+k}$ 有 $\tau_0 = 0$，故此时刻的函数与导函数值是

$$y_{n+k} = d_0$$

$$\dot{y}_{n+k} = -\frac{1}{h_k} d_1$$

这意味着，如果从当前时刻以新的步长来计算，τ_j 值必须重新计算。从而可得$(k-1)$的预估公式及校正公式分别由下面的矩阵方程来确定

预估公式

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \tau_2 & \tau_2^2 & \cdots & \tau_2^{k-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \tau_k & \tau_k^2 & \cdots & \tau_k^{k-1} \end{bmatrix} \begin{bmatrix} d_0^P \\ d_1^P \\ \vdots \\ d_{k-1}^P \end{bmatrix} = \begin{bmatrix} y_{n+k-1} \\ y_{n+k-2} \\ \vdots \\ y_n \end{bmatrix} \tag{3.69}$$

校正公式

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \tau_{k-1} & \tau_{k-1}^2 & \cdots & \tau_{k-1}^{k-1} \end{bmatrix} \begin{bmatrix} d_0^C \\ d_1^C \\ \vdots \\ d_{k-1}^C \end{bmatrix} = \begin{bmatrix} y_{n+k} \\ y_{n+k-1} \\ \vdots \\ y_{n+1} \end{bmatrix} \tag{3.70}$$

可得相应的表达式为

$$\left.\begin{aligned} y_{n+k} &= d_0^P = \sum_{j=1}^{k} \alpha_j^P y_{n+k-j} \\ \dot{y}_{n+k} &= -\frac{1}{h_k} d_1^C = -\frac{1}{h_k} \sum_{j=0}^{k-1} \alpha_j^C y_{n+k-j} \end{aligned}\right\} \tag{3.71}$$

注意，虽然前若干步的值没有变，但其系数矩阵每步均要变化。

在固定步长时，我们已经给出了误差估计，那么在调整步长时，误差如何估计呢？Brayton 在文献[9,10]中比较了各种后向差分公式，给出了如下公式

$$E = \frac{h_k (y_{n+k}^C - y_{n+k}^P)}{\alpha_0^C (t_{n+k} - t_n)} = \frac{h_k D}{\alpha_0^C T} \tag{3.72}$$

图 3.7 对该公式的含义进行了图示。

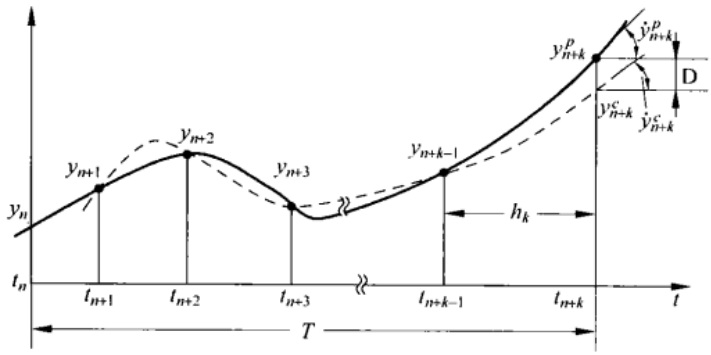


图 3.7 变步长的多步法公式的误差估计

注意，图中表示的是变步长的，y_{n+k}^C是迭代收敛时用校正公式获得的结果，y_{n+k}^P是同阶的预估公式获得的结果。α_0^C是前述校正公式$\dot{y}_{n+k}$的表达式(式(3.71))的第一个系数。

选取下一个步长的方法很多，这里介绍其中一种。一般说来，用户知道总的仿真时间t_{end}和总的允许误差，若定义单位时间内的允许误差为

$$\varepsilon = \frac{\text{总的允许误差}}{t_{\text{end}}}$$

用ε值来监控每一步误差。假设上一步的步长为h_{old}，估计误差为E_{old}，则可令

$$E_{\text{new}} \approx h_{\text{new}}\varepsilon$$

由式(3.64)，可知新旧两个步长的泰勒展开式的截断误差是

$$E_{\text{old}} \approx h_{\text{old}}^k C_k y^{(k)}(\theta)/k!$$

$$E_{\text{new}} \approx h_{\text{new}}^k C_k y^{(k)}(\theta)/k!$$

其中θ是$[t_n, t_{n+k}]$内的某一点。假设y的第k阶导数变化缓慢，两式相除，可得

$$E_{\text{new}} \approx \left(\frac{h_{\text{new}}}{h_{\text{old}}}\right)^k E_{\text{old}}$$

从而可得

$$\varepsilon h_{\text{new}} \approx \left(\frac{h_{\text{new}}}{h_{\text{old}}}\right)^k E_{\text{old}}$$

若记$\eta = h_{\text{new}}/h_{\text{old}}$，则不难得到

$$h_{\text{new}} = \eta^k E_{\text{old}}/\varepsilon$$

当求解多个方程时，必须要考虑满足每个方程求解的精度要求，即选择最小的$\eta_j = |\varepsilon h_{\text{old}}/E_{\text{old},j}|^{1/(k+1)}$，从而新步长为

$$h_{\text{new}} = (\min\eta_j) h_{\text{old}}$$

3.4 稳定性分析

利用数值积分法进行仿真时常常会发现这样的现象，本来是稳定的系统，仿真结果却得出不稳定的结论。这种现象通常是由于计算步长选得太大而造成的，因为当步长h选得过大时，计算误差较大，数值积分法会使各种误差传播出去，以致引起计算不稳定。

关于计算的稳定性可以这样理解。大家知道，实际计算中，给出的初值 $y(t_0)=y_0$ 不一定很准(即有初始误差)；同时，由于计算机字长有限，在计算中会有舍入误差；特别是，在某一步长 h 下存在截断误差，所有这些误差都可能在计算中传播下去。

下面先以一阶显式阿达姆氏法为例，讨论数值积分法稳定域问题。

假定系统的微分方程为

$$\mathrm{d}y/\mathrm{d}t=\mu y,\qquad \mu=\alpha+\mathrm{j}\beta \tag{3.73}$$

且 $\mathrm{Re}\mu=\alpha<0$，即原系统是稳定的。若用一阶阿达姆氏法来对它进行仿真，则有如下的递推公式

$$y_{k+1}=y_k+hf(t_k,y_k)$$

故有

$$y_{k+1}=y_k+h\mu y_k \tag{3.74}$$

今假定 $y_k(k=0,1,2,\cdots)$为它的一个仿真解，另外设 $y_k+\varepsilon_k$ 为其准确解，即

$$(y_{k+1}+\varepsilon_{k+1})=(y_k+\varepsilon_k)+h\mu(y_k+\varepsilon_k) \tag{3.75}$$

用式(3.75)减去式(3.74)，可得

$$\varepsilon_{k+1}=\varepsilon_k+h\mu\varepsilon_k$$

即

$$\varepsilon_{k+1}-(1+\mu h)\varepsilon_k=0 \tag{3.76}$$

其特征方程为

$$z-(1+\mu h)=0 \tag{3.77}$$

显然，为了使扰动序列 ε_k 不随 k 增加而增长，必须要求它的特征根 $z=1+\mu h$ 在单位圆内，即

$$|1+\mu h|\leqslant 1 \tag{3.78}$$

我们称它所对应的域就是该算法的稳定域，如图 3.8 所示。

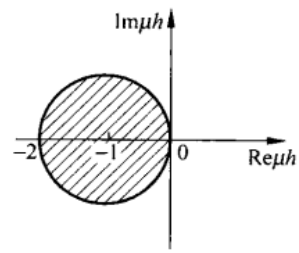


图 3.8 一阶显式阿达姆氏法稳定域

由图 3.8 可知，若系统方程的 $\mu=\alpha,\alpha<0$，采用一阶显式阿达姆氏法，为保证计算稳定，要求 $h\leqslant 2|1/\alpha|$，即要求 h 小于等于系统时间常数的两倍。

由上例可以得到确定数值积分法稳定域的一般方法。设系统方程为式(3.67)(通常该方程称为测试方程)，而数值积分公式为

$$y_{k+1}=p(h\mu)\cdot y_k \tag{3.79}$$

其中 $p(h\mu)$是一个关于 $h\mu$ 的高阶多项式函数，则只有当 $|p(h\mu)|<1$ 时，算法才稳定。根据不同算法的表达式，可在 $h\mu$ 域上画出它们的稳定域来。

图 3.9 为本节中介绍的龙格-库塔法及阿达姆氏法的稳定域。由图 3.9 可知：

(1) 除隐式一阶、二阶阿达姆氏法为恒稳法外，其它方法都是条件稳定的；

(2) 除恒稳法外，其它方法的 h 都应限制在最少时间的数量级；

(3) 对龙格-库塔法，k 大则稳定域略微增大，而对阿达姆氏法来讲，k 大则稳定域反而缩小。

需要指出的是，对于龙格-库塔法，图 3.9 所示之稳定域只适合于阶次 k 与级数 s 相同的情况，而当 $k\neq s$ 时，同一阶的各种方法的稳定域略有不同。

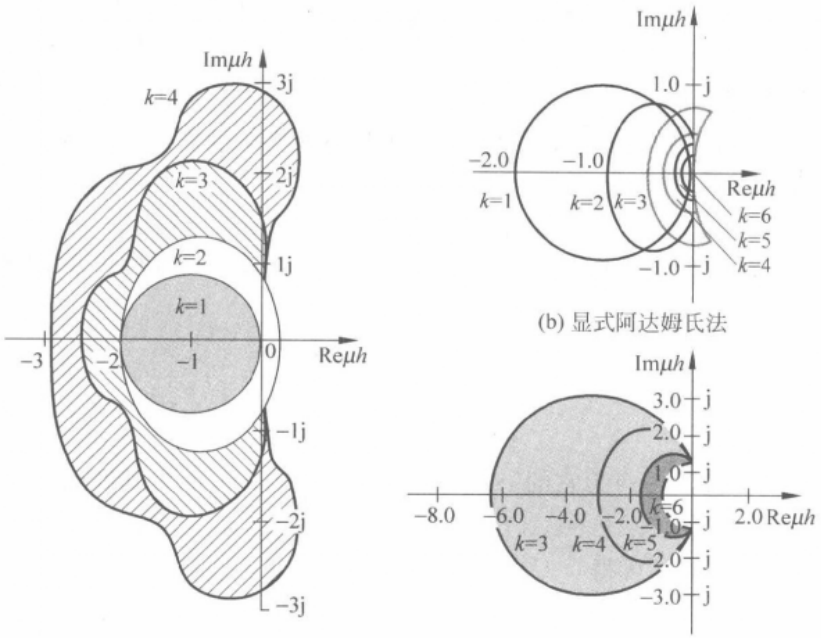


图 3.9 各种数值积分的稳定域

3.5 病态系统仿真*

本节首先就病态系统加以定义，然后对病态系统的仿真方法进行一般性讨论。如果系统是线性的，则可采用某些特殊的方法进行有效的仿真，而对于一般病态系统仿真，则必须非常谨慎地选择算法。本节介绍四种典型病态系统仿真算法，即吉尔(Gear)法、单步多值法、半隐式龙格-库塔法、多帧速算法。接下来就间断特性的仿真建模进行讨论。

3.5.1 病态系统的定义

在前面几节中，我们对连续系统的一般仿真建模方法进行了讨论。然而，在仿真实践中我们经常会碰到这样一种情况：系统中各环节的时间常数差异巨大。为保证仿真计算的稳定性，由于仿真步长必须限制在最小时间常数的数量级而选得很小，然而仿真结束的时间则决定于系统中的最大时间常数，若按满足稳定性要求所选择的步长进行仿真，则不仅整个仿真所花费的时间非常长，甚至由于计算的舍入误差而导致整个仿真的失败。这就是所谓“病态系统仿真”问题。

下面首先让我们对病态系统加以定义。若系统的动态特性可用如下形式的一阶微分方程组来描述

$$\dot{\boldsymbol{y}} = \boldsymbol{F}(\boldsymbol{y}, t) \tag{3.79}$$

其中

$$\boldsymbol{y}(t_0) = \boldsymbol{y}(0) \tag{3.80}$$

$$\boldsymbol{F} = (f_1, f_2, \cdots, f_n)^{\mathrm{T}} \tag{3.81}$$

$$\boldsymbol{y} = [y_1, y_2, \cdots, y_n]^{\mathrm{T}} \tag{3.82}$$

其中 T 表示转置。令

$$\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{y}} = \begin{bmatrix} \dfrac{\partial f_1}{\partial y_1} & \cdots & \dfrac{\partial f_1}{\partial y_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_n}{\partial y_1} & \cdots & \dfrac{\partial f_n}{\partial y_n} \end{bmatrix} = \boldsymbol{J} \tag{3.83}$$

$\boldsymbol{J}$ 称为系统的雅可比矩阵。若 $\boldsymbol{J}$ 的特征值 μ_i 全部具有负实部，且有

$$\min | \mu_i | \ll \max | \mu_i | \qquad (i = 1, 2, \cdots, n)$$

则该系统称为病态系统，在某些文献中也叫做刚性系统，而

$$S = \frac{\max | \mu_i |}{\min | \mu_i |}$$

称为病态比，一般在 50 以上。

病态系统在实际中是经常遇到的。例如，自动控制系统中控制回路常常是反应灵敏的，能够迅速完成状态过渡，即具有小时间常数，而受控对象本身的运动由于惯性大，状态过渡慢，即具有大时间常数。在多组分化学反应中，可能有些组分反应很快，有些则很慢。在电力网络、电子网络的传输或扩散过程中，对空间变量离散化而得到的常微分方程往往也存在量级悬殊的时间常数。

顺便说明一下，在实际系统中往往有这样的元件或环节，它对全过程并无实际性的影响，但却带来极小的时间常数，导致系统病态化，给仿真计算带来极大的累赘。在对这类系统建模时，应该适当地进行处理，以避免不必要的病态性。

由于线性系统具有一定的特殊性，下面首先讨论线性病态系统的仿真方法，然后再讨论非线性病态系统的仿真方法。

3.5.2 线性病态系统仿真

对线性定常系统，我们可用如下状态方程进行一般性描述

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) \\ \boldsymbol{x}(t_0) = \boldsymbol{x}(0) \end{cases} \tag{3.84}$$

1. 增广矩阵法在线性病态系统中的应用

若将 $\boldsymbol{u}(t)$ 作为系统的增广状态，将原状态方程变换为齐次状态方程，从而得到新的输出方程和新的增广状态初值

$$\begin{cases} \dot{\tilde{\boldsymbol{x}}} = \tilde{\boldsymbol{A}}\,\tilde{\boldsymbol{x}}(t) \\ \boldsymbol{y}(t) = \tilde{\boldsymbol{C}}\,\tilde{\boldsymbol{x}}(t) \\ \tilde{\boldsymbol{x}}(t_0) = \tilde{\boldsymbol{x}}(0) \end{cases} \tag{3.85}$$

其中上带波浪线的变量表示变换后的增广变量。采用离散相似法进行仿真，则可得到增广状态方程的仿真模型

$$\tilde{\boldsymbol{x}}(k+1)=\tilde{\boldsymbol{\phi}}(t)\,\tilde{\boldsymbol{x}}(k) \tag{3.86}$$

其中

$$\tilde{\boldsymbol{\phi}}(T)=\mathrm{e}^{\tilde{\boldsymbol{A}}h}$$

为增广状态转移矩阵，h 为仿真步长。然而，对于病态系统，由于其特征值相差倍数很大，必须用第4章中介绍的加速收敛的方法来计算该状态转移矩阵的值。

2. 蛙跳算法

加速收敛计算增广状态转移矩阵时对仿真步长没有严格的限制，因而对仿真速度不会带来显著的影响，而且只要增广状态转移矩阵的计算是收敛的，则不失仿真的稳定性。然而，对于常见的阶跃函数输入下的过渡过程的仿真计算来说，若仿真步长 h 选得过大(相对最小时间常数而言)，初始阶段的仿真结果误差就会加大。

为了提高初始阶段仿真结果的精度，人们提出一种所谓“蛙跳算法”，其基本思想是：

(1) 首先将式(3.84)的状态方程转换为式(3.85)，考虑作用函数为阶跃函数，则增广状态十分简单。

(2) 用 Pade 表达式来逼近增广状态转移矩阵，即

$$\mathrm{e}^{\boldsymbol{A}h}\approx\left(1-\frac{1}{2}\boldsymbol{A}h+\frac{1}{12}\boldsymbol{A}^2h^2\right)^{-1}\cdot\left(1+\frac{1}{2}\boldsymbol{A}h+\frac{1}{12}\boldsymbol{A}^2h^2\right) \tag{3.87}$$

(3) 选择 q，使得 $q=2^n$，$qh=2^nh$，仿真计算时采用如下“蛙跳”方式

$$\begin{aligned}
&\boldsymbol{x}(h)=\mathrm{e}^{\boldsymbol{A}h}\boldsymbol{x}(0)\\
&\boldsymbol{x}(2h)=\mathrm{e}^{\boldsymbol{A}h}\cdot\mathrm{e}^{\boldsymbol{A}h}\boldsymbol{x}(0)=\mathrm{e}^{2\boldsymbol{A}h}\boldsymbol{x}(0)\\
&\boldsymbol{x}(4h)=\mathrm{e}^{2\boldsymbol{A}h}\cdot\mathrm{e}^{2\boldsymbol{A}h}\boldsymbol{x}(0)=\mathrm{e}^{4\boldsymbol{A}h}\boldsymbol{x}(0)\\
&\qquad\vdots\\
&\boldsymbol{x}(2^nh)=\boldsymbol{x}(qh)=\mathrm{e}^{2^{n-1}\cdot\boldsymbol{A}h}\cdot\mathrm{e}^{2^{n-1}\cdot\boldsymbol{A}h}\boldsymbol{x}(0)=\mathrm{e}^{2^n\cdot\boldsymbol{A}h}\boldsymbol{x}(0)=\mathrm{e}^{2^{n-1}\cdot\boldsymbol{A}h}\boldsymbol{x}(0)\\
&\boldsymbol{x}(2qh)=\mathrm{e}^{2^{n-1}\cdot\boldsymbol{A}h}\boldsymbol{x}(qh)\\
&\boldsymbol{x}(3qh)=\mathrm{e}^{2^{n-1}\cdot\boldsymbol{A}h}\boldsymbol{x}(2qh)\\
&\boldsymbol{x}(4qh)=\mathrm{e}^{2^{n-1}\cdot\boldsymbol{A}h}\boldsymbol{x}(3qh)\\
&\qquad\vdots
\end{aligned}$$

即在 qh 以前采用跳跃式计算，而在 qh 以后每隔 qh 计算一次。采用这种方法的优点是：h 可以取得很小(比如按最小时间常数考虑)，从而保证初始阶段的精度而计算量却不大，而到 qh 以后，小时间常数的作用完成，则加大步长计算，从而加快仿真计算速度。另外，从 $\boldsymbol{x}(h)$ 到 $\boldsymbol{x}(qh)$ 都是以 $\boldsymbol{x}(0)$ 为基础进行计算，所以误差传播比较均匀(仅仅是状态转移矩阵的误差)。适当选取 q 的值，就可以兼顾仿真的速度和精度，并保证仿真过程的稳定性。

以式(3.88)所示的方程为例

$$\begin{cases}\dot{\boldsymbol{y}}=\boldsymbol{A}\boldsymbol{y}\\ \boldsymbol{y}(0)=[1\quad 0\quad 1]\end{cases} \tag{3.88}$$

其中

$$\boldsymbol{A}=\begin{bmatrix}-21 & 19 & -20\\ 19 & -21 & 20\\ 40 & -40 & -40\end{bmatrix}$$

该系统的最大特征值与最小特征值绝对值之比为 20，属于轻度病态系统。其解析解为

$$y_1(t)=0.5e^{-2t}+0.5e^{-40t}(\cos 40t+\sin 40t)$$

$$y_2(t)=0.5e^{-2t}-0.5e^{-40t}(\cos 40t+\sin 40t)$$

$$y_3(t)=0.5e^{-40t}(\cos 40t-\sin 40t)$$

仿真计算表明，蛙跳法采用 $h=0.003\,125$，$qh=0.05$ 计算，而用四阶龙格-库塔法，步长为 0.002，前者的速度为后者的 24 倍，而计算精度却几乎相同。表 3.7 给出了两者与解析计算结果的比较。

表 3.7 蛙跳法与 RK-4 法比较

T	$y(t)$		
	解析解	蛙跳法	RK-4 法
0.05	0.485 789 05	0.485 789	0.485 789
0.10	0.396 448 77	0.396 449	0.396 449
0.15	0.371 252 84	0.371 253	0.371 253
0.20	0.335 301 58	0.335 302	0.335 302
0.25	0.303 233 93	0.303 234	0.303 234
0.30	0.274 406 75	0.274 407	0.274 407
0.35	0.248 293 10	0.248 293	0.248 293
0.40	0.224 664 39	0.224 665	0.224 660
0.45	0.203 284 81	0.203 285	0.203 285
0.50	0.183 939 71	0.183 940	0.183 940

3.5.3 吉尔(Gear)法

上一节我们讨论了线性病态系统仿真问题，其核心归结为状态转移矩阵的处理。然而，对非线性病态系统，则问题要复杂得多。

一般非线性系统的仿真大多采用数值积分法。而数值积分法一般又只具有有限的稳定域，典型的如龙格-库塔法，仿真步长限定在系统最小时间常数的数量级，才能保证计算的稳定性，而系统的过渡过程时间却决定于最大时间常数，因而对病态系统来说计算量极大，加上存在误差传播，仿真的精度甚至稳定性也会受到影响。具有恒稳特性的数值积分法虽然对仿真步长没有限制，但只有一阶和二阶，截断误差大，也难以用于非线性病态系统的仿真。

本节讨论用于一般非线性病态系统仿真的吉尔法。首先介绍所谓“Stiff 稳定域”，然后讨论基于满足 Stiff 稳定域的吉尔法，包括该方法的基本原理和实现策略。

(1) Stiff 稳定域

Gear 对非线性病态系统进行深入研究后发现，这类系统的仿真并不要求一定采用恒稳

方法，而只要具有所谓 Stiff 稳定域就可以了。他将 Stiff 稳定域定义为如下区域：

若某一种方法的稳定域满足图 3.10 中的阴影部分，则称满足 Stiff 稳定域。

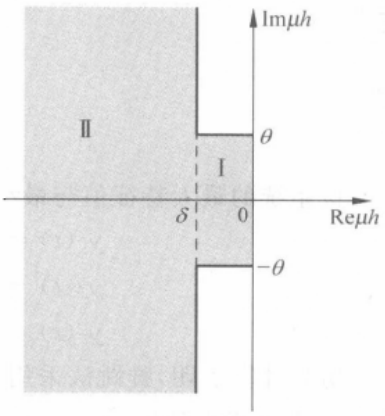


图 3.10 Stiff 稳定域

对稳定的物理系统，特征值 μ 的实部一般小于零，因而只要仿真步长 h 的选择满足

$$\begin{cases} \mathrm{Re}\mu h \leqslant 0 \\ |\ \mathrm{Im}\mu h\ | \leqslant \theta \end{cases} \qquad \text{条件(1)}$$

与

$$\mathrm{Re}\mu h \leqslant \delta \qquad \text{条件(2)}$$

即可保证仿真的稳定性，则称该算法具有 Stiff 稳定域。

实际上满足 Stiff 稳定域的方法与恒稳方法只在近虚轴处有一点差别，即如果系统中的极点全部为实极点，那么无论选择多大的步长，只要满足条件(2)，计算是恒稳的。如果系统中有复极点(实部仍为负数)，只要步长的选择满足条件(1)，也能保证算法稳定。以后我们将满足 Stiff 稳定域的方法简称为 Stiff 法。

下面我们进一步分析如何确定 Stiff 域中 θ 与 δ 的值。这要从病态系统的特点出发。如前所述，病态系统的特征值相差很大(特征值比大于 50)。为了准确地计算出系统的瞬态响应，这就要按大特征值来选择步长。设系统最大特征值为

$$\mu_{\max} = \alpha_{\max} + \mathrm{j}\beta_{\max}$$

设该特征值所对应的模态大约要经过时间常数 4 倍左右的时间才能有效地衰减掉，即

$$h \leqslant 4/|\ \alpha_{\max}\ |$$

$$h\ |\ \alpha_{\max}\ | \leqslant 4$$

这样，此时即使加大步长 h，也能保持计算的稳定性。基于这一考虑，可设 $\delta=-4$。

另一方面，考虑到系统特征值为复数，它所对应的瞬态响应呈振荡型。为保证仿真的准确性，在一个振荡周期内至少计算 N 个点。最小振荡周期为

$$T_{\min} = 2\pi/\beta_{\max} = Nh$$

其中 h 为计算步长，若选择 $N \geqslant 8$，则有

$$\beta_{\max} h \leqslant \pi/4$$

因此可选 $\theta=\pi/4$。

综上所述，如果选择某一种方法，其稳定域 $\theta \geqslant \pi/4$，且 $|\delta| \leqslant 4$，则从使用的角度来看，图 3.10 所示的稳定域与恒稳域没有差别，从而完全可以用于病态系统的仿真。

(2) 吉尔法的基本原理

基于 Stiff 稳定域的定义，Gear 首先推出了满足 Stiff 稳定域的多步法。这里，我们以一阶微分方程为例加以说明。设系统为

$$\dot{y} = \mathrm{d}y/\mathrm{d}t = f(y,t), \qquad y(t=t_0) = y_0$$

由多步法计算公式

$$\sum_{j=0}^{k} \alpha_j y_{n+k-j} - h\sum_{j=0}^{k} \beta_j\, \dot{y}_{n+k-j} = 0$$

Gear 提出的用于病态系统仿真的计算公式是

$$y_{n+k}=\sum_{j=1}^{k}\alpha_j y_{n+k-j}-h\beta_0\dot{y}_{n+k} \tag{3.89}$$

其中公式中的各个系数如表 3.8 所示。

表 3.8 用于病态系统仿真的 Gear 公式的系数表

	α_1	α_2	α_3	α_4	α_5	α_6	β_0
G-1	1						1
G-2	4/3	−1/3					2/3
G-3	18/11	−9/11	2/11				6/11
G-4	48/25	−36/25	16/25	−3/25			12/25
G-5	300/137	−300/137	200/137	−75/137	12/137		60/137
G-6	360/147	−450/147	400/147	−225/147	72/147	−10/147	60/147

可以由表达式推出其相应的稳定域如图 3.11 所示。从图上可以看出，该方法在五阶以下(包括五阶)的稳定域满足 Stiff 稳定域的条件($\theta>\pi/4$，$|\delta|<4$)，大于五阶时$|\delta|\geqslant 4$，而且还可能穿过负实轴。

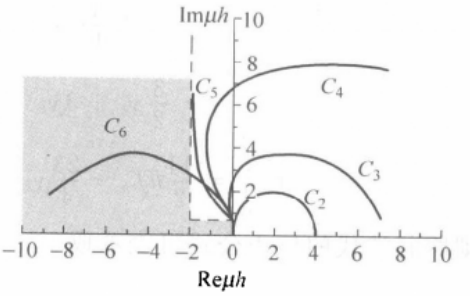


图 3.11 用于病态系统仿真的 Gear 法

在用 Gear 法仿真非线性病态系统时，有以下三个基本问题需要解决：

① 启动问题

式(3.89)所描述的 Gear 法本质上是隐式多步法。对于初值问题，多步法无法自启动，加上又是隐式法，这就更为困难。

隐式方法一般用显式方法启动，即先进行预报，然后通过迭代进行校正。如果迭代方法的收敛性不好，可能引起计算发散或计算量加大。

即使选择的迭代方法收敛性满足要求，若采用显式多步法预报，仍然难以启动，必须采用单步法启动，由于单步法不具有 Stiff 稳定域，因而很难保证计算的稳定性。

② 变步长策略

对非线性病态系统仿真往往采用变步长策略，即开始采用小步长试探，然后再逐步放大步长。如何适时地将步长调整到合适长度，以同时满足仿真精度和速度的要求，是仿真实现过程必须解决的问题。

③ 加速迭代

为了提高计算效能，加速迭代也是非线性病态系统仿真中的重要问题。

下面我们就上述三方面的问题进行进一步讨论。

3.5.4 单步多值法

为了导出单步多值法，让我们先看一个实际的例子，采用显式多步法进行预报，然后用隐式法校正。以三阶为例，其显式预报的公式是

$$y_{k+1}^{(0)} = -\frac{3}{2}y_k + 3y_{k-1} - \frac{1}{2}y_{k-2} + 3hf_k \tag{3.90}$$

其中带括弧的上标表示第几次迭代(以下相同)。再用三阶隐式 Gear 公式校正

$$\begin{aligned} y_{k+1}^{(1)} &= \frac{18}{11}y_k - \frac{9}{11}y_{k-1} + \frac{2}{11}y_{k-2} + \frac{6}{11}hf_{k+1}^{(0)} \\ y_{k+1}^{(2)} &= \frac{18}{11}y_k - \frac{9}{11}y_{k-1} + \frac{2}{11}y_{k-2} + \frac{6}{11}hf_{k+1}^{(1)} \\ &\vdots \end{aligned} \tag{3.91}$$

其中等式右边第 4 项为导函数项,上标为 i 的导函数值为预报值,它是通过将第 i 次迭代所得到的 y 的预报值和已知的 y 值一同代入导函数后计算得到的。为便于程序实现,由式(3.90),并令

$$\begin{aligned} y_{k+1}^{(1)} &= \frac{18}{11}y_k - \frac{9}{11}y_{k-1} + \frac{2}{11}y_{k-2} + \frac{6}{11}hf_{k+1}^{(0)} \\ &= y_{k+1}^{(0)} + \frac{6}{11}(hf_{k+1}^{(0)} - f_{k+1}^{(-1)}) \\ &= -\frac{3}{2}y_k + 3y_{k-1} - \frac{1}{2}y_{k-2} + 3hf_k + \frac{6}{11}(hf_{k+1}^{(0)} - hf_{k+1}^{(-1)}) \\ f_{k+1}^{(-1)} &= \frac{11}{2}hf_k - \frac{23}{4}y_k + 7y_{k-1} - \frac{5}{4}y_{k-2} \end{aligned}$$

则所有迭代的校正公式可表示成

$$\begin{aligned} y_{k+1}^{(1)} &= y_{k+1}^{(0)} + \frac{6}{11}(hf_{k+1}^{(0)} - hf_{k+1}^{(-1)}) \\ y_{k+1}^{(2)} &= y_{k+1}^{(1)} + \frac{6}{11}(hf_{k+1}^{(1)} - hf_{k+1}^{(0)}) \\ &\vdots \\ y_{k+1}^{(i+1)} &= y_{k+1}^{(i)} + \frac{6}{11}(hf_{k+1}^{(i)} - hf_{k+1}^{(i-1)}) \\ &\vdots \end{aligned}$$

对一般情形,再进一步,我们令

$$\begin{aligned} \boldsymbol{y}_k &= (y_k, hf_k, y_{k-1}, y_{k-2}, \cdots, y_{k-p+1})^{\mathrm{T}} \\ \boldsymbol{y}_{k+1} &= (y_{k+1}, hf_{k+1}, y_k, y_{k-1}, \cdots, y_{k-p+2})^{\mathrm{T}} \\ G_{k+1}^{(i)} &= hf_{k+1}^{(i)} - f_{k+1}^{(i-1)} \\ &= hf(y_{k+1}^{(i)}, t_{k+1}) - hf(y_{k+1}^{(i-1)}, t_{k+1}) \end{aligned}$$

其中 p 表示 Gear 法的阶次。对三阶预报-迭代校正算法,可得到

$$\begin{bmatrix} y_{k+1}^{(0)} \\ hf_{k+1}^{(-1)} \\ y_k \\ y_{k-1} \end{bmatrix} = \begin{bmatrix} -\frac{3}{2} & 3 & 3 & -\frac{1}{2} \\ -\frac{23}{4} & \frac{11}{2} & 7 & -\frac{5}{4} \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} y_k \\ hf_k \\ y_{k-1} \\ y_{k-2} \end{bmatrix}$$

$$\begin{bmatrix} y_{k+1}^{(i+1)} \\ hf_{k+1}^{(i)} \\ y_k \\ y_{k-1} \end{bmatrix} = \begin{bmatrix} y_{k+1}^{(i)} \\ hf_{k+1}^{(i-1)} \\ y_k \\ y_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{6}{11} \\ 1 \\ 0 \\ 0 \end{bmatrix} G_{k+1}^{(i)}$$

则可将上述预报与迭代校正表达式用矩阵的方法加以表示

$$\boldsymbol{y}_{k+1}^{(0)} = \boldsymbol{B}\boldsymbol{y}_k \qquad (i = 1,2,\cdots,k-1) \tag{3.92}$$

$$\boldsymbol{y}_{k+1}^{(i+1)} = \boldsymbol{y}_{k+1}^{(i)} + \boldsymbol{C}\boldsymbol{G}_{k+1}^{(i)} \qquad (i = 1,2,\cdots,k-1) \tag{3.93}$$

其中 $\boldsymbol{B},\boldsymbol{C}$ 分别是相应的系数矩阵。然而，对初值问题，上述表达式是不能自启动的。

为解决这一问题，人们提出单步多值法以避免传统的单步法所带来的局限性。其基本思想是用上一步的高阶导数值来取代前几步的 y 及 f 的值。为此，我们先定义一个向量，称为 Nordsieck 向量

$$\boldsymbol{z}_k = \left(y_k, hy_k, \frac{h^2}{2!}y_k^{(2)}, \cdots, \frac{h^p}{p!}y_k^{(p)}\right)^{\mathrm{T}} \tag{3.94}$$

$$\boldsymbol{z}_{k+1} = \left(y_{k+1}, hy_{k+1}, \frac{h^2}{2!}y_k^{(2)}, \cdots, \frac{h^{p+1}}{(p+1)!}y_{k+1}^{(p+1)}\right)^{\mathrm{T}} \tag{3.95}$$

现在需要确定 $\boldsymbol{z}$ 向量与 $\boldsymbol{y}$ 向量之间的关系。采用多项式逼近来拟合，以三阶为例

$$\begin{cases} y(t) = a_0 + a_1(t-t_k) + a_2(t-t_k)^2 + a_3(t-t_k)^3 \\ \dot{y}(t) = a_1 + 2a_2(t-t_k) + 3a_3(t-t_k)^2 \\ \ddot{y}(t) = 2a_2 + 6a_3(t-t_k) \\ \dddot{y}(t) = 6a_3 \end{cases} \tag{3.96}$$

从而在 $t=t_k$ 处，有

$$\begin{cases} y_k = a_0 \\ f_k = \dot{y}_k = a_1 \\ y_{k-1} = y(t_k - h) = a_0 + a_1(-h) + a_2h^2 + a_3(-h)^3 \\ y_{k-2} = y(t_k - 2h) = a_0 + a_1(-2h) + a_2(-2h)^2 + a_3(-2h)^3 \end{cases} \tag{3.97}$$

同样，也可以得到

$$y_k = a_0, \quad \dot{y}_k = a_1, \quad \ddot{y}_k = 2!a_2, \quad \dddot{y}_k = 3!a_3$$

由式(3.96)及式(3.97)消去 a_0,a_1,a_2,a_3，整理后可得到

$$\begin{cases} y_k = y_k \\ h\dot{y}_k = hf_k \\ \frac{h^2}{2!}\ddot{y}_k = -\frac{7}{4}y_k + \frac{3}{2}hf_k + 2y_{k-1} - \frac{1}{4}y_{k-2} \\ \frac{h^3}{3!}\dddot{y}_k = -\frac{3}{4}y_k + \frac{13}{2}hf_k + y_{k-1} - \frac{1}{4}y_{k-2} \end{cases}$$

写成矩阵形式，就是

$$\begin{bmatrix} y_k \\ h\dot{y}_k \\ \frac{h^2}{2!}\ddot{y}_k \\ \frac{h^3}{3!}\dddot{y}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{7}{4} & \frac{3}{2} & 2 & -\frac{1}{4} \\ -\frac{3}{4} & \frac{13}{2} & 1 & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} y_k \\ hf_k \\ y_{k-1} \\ y_{k-2} \end{bmatrix}$$

简记为

$$\boldsymbol{z}_k = \boldsymbol{Q}\boldsymbol{y}_k \tag{3.98}$$

$\boldsymbol{Q}$ 阵就是 $\boldsymbol{y}$ 向量与 $\boldsymbol{z}$ 向量之间的变换阵。对 p 阶 Gear 法，$\boldsymbol{Q}$ 阵为$(p+1)$阶非奇异方阵。

将式(3.98)代入式(3.91)及式(3.93)，可得到单步多值法的计算公式

$$\begin{cases} \boldsymbol{z}_{k+1}^{(0)} = \boldsymbol{B}\boldsymbol{Q}^{-1}\boldsymbol{z}_k \\ \boldsymbol{z}_{k+1}^{(i+1)} = \boldsymbol{z}_{k+1}^{(i)} + \boldsymbol{Q}\boldsymbol{C}\boldsymbol{G}_{k+1}^{(i)} \\ \boldsymbol{G}_{k+1}^{(i)} = hf(\boldsymbol{z}_{k+1}^{(i)}) - hf(\boldsymbol{z}_{k+1}^{(i-1)}) \end{cases} \tag{3.99}$$

需要说明的是，式(3.99)中计算 $\boldsymbol{G}_{k+1}^{(i)}$ 时以 $\boldsymbol{z}_{k+1}^{(i)}$，$\boldsymbol{z}_{k+1}^{(i-1)}$ 为自变量，但由于 $\boldsymbol{G}_{k+1}^{(i)}$ 是一个标量，所以实际上只是用到它们的第一个分量 $y_{k+1}^{(i)}$，$y_{k+1}^{(i-1)}$。

若记

$$\begin{cases} \boldsymbol{P} = \boldsymbol{Q}\boldsymbol{B}\boldsymbol{Q}^{-1} \\ \boldsymbol{L} = \boldsymbol{Q}\boldsymbol{C} \end{cases} \tag{3.100}$$

则式(3.99)可简写为

$$\begin{cases} \boldsymbol{z}_{k+1}^{(0)} = \boldsymbol{P}\boldsymbol{z}_k \\ \boldsymbol{z}_{k+1}^{(i+1)} = \boldsymbol{z}_{k+1}^{(i)} + \boldsymbol{L}\boldsymbol{G}_{k+1}^{(i)} \end{cases} \tag{3.101}$$

在三阶的情况下，$\boldsymbol{P}$ 及 $\boldsymbol{L}$ 的值如下

$$\boldsymbol{P} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{L} = \begin{bmatrix} \frac{6}{11} & 1 & \frac{6}{11} & \frac{1}{11} \end{bmatrix}^{\mathrm{T}}$$

采用同样的思想，我们可以计算得到 Gear 法的 $\boldsymbol{L}$ 向量值如表 3.9 所示。

表 3.9 一阶～五阶吉尔法 $\boldsymbol{L}$ 向量值

阶次	1	2	3	4	5
$\boldsymbol{L}0$	1	2/3	6/11	12/25	60/137
$\boldsymbol{L}1$	1	1	1	1	1
$\boldsymbol{L}2$		1/3	6/11	7/10	225/274
$\boldsymbol{L}3$			1/11	1/5	85/274
$\boldsymbol{L}4$				1/50	15/274
$\boldsymbol{L}5$					1/274

稍加观察 $\boldsymbol{P}$ 阵，不难发现，它是一个上三角阵，该三角阵的元素由二项式系数组成，这种矩阵称为 Pascal 三角阵。可以证明，对任意阶 Gear 法的 $\boldsymbol{P}$ 阵都具有该性质。

由于 $\boldsymbol{z}_k$ 为 y_k，$\dot{y}_k$，$\ddot{y}_k$，…所构成的向量，而 $\boldsymbol{z}_{k+1}$ 为 y_{k+1}，$\dot{y}_{k+1}$，$\ddot{y}_{k+1}$，…所构成的向量，因

此，式(3.101)是一种单步多值法，它可以自启动，不但克服了多步法不能自启动的困难，而且易于变步长。与一般单步法的区别在于，对每个计算步，不仅要保留变量值，而且要计算和保留其高阶导数$\dot{y}_{k+1}$，$\ddot{y}_{k+1}$，…，$y_{k+1}^{(\rho)}$值。

3.5.5 病态探测与误差估计

病态系统仿真的复杂性在于，在仿真过程中，并不是系统始终处于“病态”。显然只有当系统处于“病态”时应采用求解病态问题的方法，而处于非病态时应采用常规方法，因为后者比前者效率会高得多。然而，系统在仿真的每一步是否处于病态，不仅与系统本身有关，而且仿真算法及步长选择是相关的。归根结底，这依赖于每步仿真计算的误差判断，以保证计算的稳定性和计算精度的要求。本节将讨论病态系统仿真的三个基本问题，即病态性探测、误差估计以及加速收敛问题。这些方法和原则也适用于一般系统的仿真。

(1) 病态性探测

上面我们对吉尔法用于病态系统的仿真的有关原理及其实现技术进行了比较深入的讨论。实践证明，这种方法对病态系统的仿真是十分有效的。然而，对病态系统进行仿真的过程中，系统并不是每一步均处于病态。如果系统处于非病态而仍然采用上述方法，则会由于Gear法的大量附加计算(如计算雅可比矩阵、矩阵求逆等)而使仿真效能大大降低。

为了进一步提高仿真计算的效能，下面我们讨论仿真过程中的病态性探测问题。为此，我们首先对“病态性”和“非病态性”加以定义。这里我们是从数值计算的角度进行定义的。

如果用非Stiff法(如显式龙格-库塔法，Adams法等)对一个系统进行仿真，其步长限制是由于计算精度原因引起的而不是由于稳定性引起的，则称系统呈现“非病态性”；反之，若步长是因稳定性而受到限制时，则系统呈现“病态性”。

显然，当系统呈现病态性时应采用Stiff法，而当系统呈现非病态性时应采用非Stiff法。因此，在仿真过程中，需要适当地进行病态性探测。下面介绍常用的三种方法。

① 稳定半径法

从3.4节的讨论中我们已经知道，非Stiff法(如龙格-库塔法，Adams法)的稳定域的形状接近半圆，并将其稳定域与实轴的交点称为稳定半径(r_k)。若记ρ为系统的最大特征值，为保证计算稳定，则要求

$$\rho \cdot h \leqslant r_k \tag{3.102}$$

由于各种方法的稳定半径是已知的，而ρ的值可由雅可比矩阵的范数$\| J \|$来估计，即

$$\rho \approx \| J \|$$

因此，在用Stiff方法对病态系统仿真过程中，由于雅可比矩阵的值是可以得到的，则在下一步计算时，先用式(3.102)判断病态性，再决定是否仍采用Stiff方法仿真。

② 嵌入低阶大稳定域法

如果当前采用的是非Stiff法，而计算误差不能满足要求，为判断其稳定性，可降低阶次，如果此时误差减少，则说明当前已处于非稳定状态(非Stiff法的低阶的稳定域大于高阶的稳定域)，因而下一步应采用Stiff法计算。

③ 小稳定域试探法

嵌入低阶大稳定域法对于高阶的非Stiff法是有效的，然而，如果当前使用的已经是低

阶方法，则很难再嵌入一个更低阶的算法。小稳定域试探法采用与嵌入低阶大稳定域法相反的思路来进行病态性探测。

若当前采用的是非 Stiff 法，为探测下一步的病态性，用比其高一阶的方法相同步长进行探测，若误差加大，说明当前已处于病态或病态边缘，因此下一步要么减少步长，要么采用 Stiff 方法计算。

(2) 误差估计

当用多步的 Gear 法对非线性病态系统进行仿真时，它要求从初值开始，必须依靠显式法来启动，即先从 y_0 及 $\dot{y}_0$ 开始，按一阶公式计算，然后逐次升阶，计算出 $\ddot{y}_0, \dddot{y}_0, \cdots, y_0^{(p)}$。在这种升阶过程中必须满足误差要求，以保证仿真计算的精度。在仿真计算的过程中，为提高仿真速度，也需要不断地调整步长和阶次，以达到满足精度条件下的高速度。因此，在仿真实现时，误差估计及其控制问题必须加以解决。

从多步法讨论中我们知道，对 p 阶多步法，其截断误差为

$$E = C_{p+1} h^{p+1} y^{(p+1)}(\xi) \tag{3.103}$$

其中 $y^{(p+1)}(\xi)$ 是 $t=\xi$ 点上 $y(t)$ 的 $(p+1)$ 阶导数值，而 ξ 是所讨论区间中的某一个点。例如，对三阶 Gear 法，ξ 就是 $[t_k-2h, t_k+h]$ 区间上的某一个点。若设 $y^{(p+1)}(\xi)=y_k^{(p+1)}$，则第 k 步的截断误差为

$$E_k = C_{p+1} h^{p+1} y_k^{(p+1)} \tag{3.104}$$

为了估计 E_k，首先要估计 $y_k^{(p+1)}$，已知

$$\begin{aligned} z_k &= \left(y_k, h\dot{y}_k, \frac{h^2}{2!}y_k^{(2)}, \cdots, \frac{h^p}{p!}y_k^{(p)}\right)^{\mathrm{T}} \\ &= (z_k, z_k^1, \cdots, z_k^p)^{\mathrm{T}} \end{aligned}$$

现在要用 $y_k^{(p)}$ 的差分来近似估计 $y_k^{(p+1)}$，即

$$y_k^{(p+1)} \approx \frac{y_k^{(p)} - y_{k-1}^{(p)}}{h}$$

上式两边同乘以 $\dfrac{h^{p+1}}{p!}$，则可得

$$\begin{aligned} \frac{h^{p+1}y_k^{p+1}}{p!} &\approx \frac{y_k^{(p)} - y_{k-1}^{(p)}}{h}\frac{h^{p+1}}{p!} = \frac{h^p}{p!}y_k^{(p)} - \frac{h^p}{p!}y_{k-1}^{(p)} \\ &= z_k^p - z_{k-1}^p \\ &= \nabla z_k^p \end{aligned} \tag{3.105}$$

将式(3.105)代入式(3.104)可得

$$E_k \approx C_{p+1}\, p!\, \nabla z_k^p \tag{3.106}$$

由式(3.106)可知，误差估计值可由第 k 步计算所得到的 z_k 与前一步计算得到的 z_{k-1} 这两个向量的最后一个分量 z_k^p 与 z_{k-1}^p 相减后，按式(3.106)计算得到。

仿真中一般采用的是相对误差，若要求每一步的相对误差不大于 ε_0，即

$$C_{p+1}\, p! \left|\frac{\nabla z_k^p}{y_{\max}}\right| \leqslant \varepsilon_0 \tag{3.107}$$

其中 $y_{\max}$ 为到目前为止已出现过的 y 的最大绝对值(注：若 y 的初值为 0，则应取 $y_{\max}=1$ 为宜)。

若系统为微分方程组，状态变量 y 为 N 个 $(y(1), y(2), \cdots, y(N))$，则相对误差可定

义为

$$E_k = \sqrt{\sum_{j=1}^{N}\left[\frac{C_{p+1}\,p!\nabla z_k^p(j)}{y(j)_{\max}}\right]^2} \tag{3.108}$$

若 $E_k \leqslant \varepsilon_0$，则本步计算结果有效，进入下一步；如果不满足，则需要减小步长或者采取其他措施。

下面我们进一步讨论误差控制的有关措施。误差控制一般从两方面进行，其一是改变步长，其二是改变阶次，因为两者均对误差有直接的影响。实际上，当某一步仿真结果得到后，经误差检验，无论是 $E_k \leqslant \varepsilon_0$，$E_k > \varepsilon_0$，或 $E_k \ll \varepsilon_0$，均需要考虑变阶或(与)变步长的问题，即通过改变 p 及 h 以使 $E_k \approx \varepsilon_0$。下面讨论变阶或(与)变步长的原则。

一般，首先考虑的是仅改变步长 h(阶次不变)，设新的步长为 h_s，且 $h_s = R_p h$。那么，R_p 应取多大为宜呢？为简便起见，我们根据单变量表达式(3.104)来分析，即

$$\left|\frac{C_{p+1}y_k^{p+1}\cdot(R_p h)^{p+1}}{y_{\max}}\right| \approx \varepsilon_0$$

$$R_p^{p+1}\left|\frac{C_{p+1}y_k^{p+1}\cdot h^{p+1}}{y_{\max}}\right| \approx \varepsilon_0$$

根据式(3.105)，$h^{p+1}y_k^{(p+1)} = p!\ \nabla z_k^p$，故有

$$R_p^{p+1}\left|\frac{C_{p+1}\,p!\nabla z_k^p}{y_{\max}}\right| \approx \varepsilon_0$$

这样，可得到

$$R_p \approx \sqrt[p+1]{\frac{\varepsilon_0}{C_{p+1}\,p!\dfrac{\nabla z_k^p}{y_{\max}}}}$$

考虑到误差仅仅是估计值，为保险起见，上式所得到的 R_p 应乘一安全系数为宜，一般该安全系数取值为 1/1.2，这样，R_p 表达式为

$$R_p = \frac{1}{1.2}\sqrt[p+1]{\frac{\varepsilon_0}{C_{p+1}\,p!\left|\dfrac{\nabla z_k^p}{y_{\max}}\right|}} \tag{3.109}$$

如果当前仿真结果误差远大于规定误差的要求，改变步长的效果也不理想，则考虑要提升仿真方法的阶次。设当前为 p 阶，考虑用 $p+1$ 阶计算。假设此时步长 $h_s = R_{p+1}h$ 能使相对误差接近规定的要求，即

$$\left|\frac{C_{p+2}\cdot R_{p+1}^{p+2}h^{p+2}y_k^{(p+2)}}{y_{\max}}\right| \approx \varepsilon_0$$

采用二阶差分来近似 $y_k^{(p+2)}$，即

$$y_k^{(p+2)} = \frac{y_k^{(p+1)} - y_{k-1}^{(p+1)}}{h}$$

考虑到

$$y_k^{(p+1)} = \frac{h^p}{p!}\nabla z_k^p$$

则不难得到

$$y_k^{(p+2)} = \frac{p!}{h^{p+1}}\frac{\nabla z_k^p - \nabla z_{k-1}^p}{h} = \frac{p!}{h^{p+2}}\nabla^2 z_k^p$$

将其代入误差估计表达式，若也考虑安全系数（这里取其为1/1.4，这是经验系数），则可得到升阶时 R_{p+1} 的表达式

$$R_{p+1}=\frac{1}{1.4}\sqrt[p+2]{\frac{\varepsilon_0}{C_{p+2}p!\left|\frac{\nabla^2 z_k^p}{y_{\max}}\right|}} \tag{3.110}$$

如果 $E_k \ll \varepsilon_0$，则说明仿真的阶次可以不必那么高，为提高仿真速度，可降低阶次。降阶可能发生的另一种情况是：由于高阶次方法的稳定域低于低阶次方法的稳定域，如果仿真步长缩短到相当小而其误差仍然达不到要求，则可能是因阶次过高而稳定域达不到要求的缘故。若将当前仿真的阶次 p 降低一阶，仿真步长 h 变为 $h_s=R_{p-1}h$，此时要使仿真误差接近规定要求，则

$$\left|\frac{C_p \cdot R_{p-1}^p h^p y_k^{(p)}}{y_{\max}}\right| \approx \varepsilon_0$$

同样，根据经验，考虑安全系数为1/1.3，则可得到

$$R_{p-1}=\frac{1}{1.3}\sqrt[p]{\frac{\varepsilon_0}{C_p p!\left|\frac{z_k^p}{y_{\max}}\right|}} \tag{3.111}$$

式中 $z_k^p=\frac{h^p}{p!}y_k^{(p)}$，它是 z_k 向量的最后一个分量。

由于积分过程中，阶次和步长均可能发生变化，特别是步长发生变化时，如何由当前的 z_k 产生在 h_s 下的 z_{ks}，这也是在仿真实现时必须解决的问题。下面也按上述三种情况加以讨论。

已知 $z_k=(y_k, h\dot{y}_k, \frac{h^2}{2!}y_k^{(2)}, \cdots, \frac{h^p}{p!}y_k^{(p)})^{\mathrm{T}}=(z_k, z_k^1, \cdots, z_k^p)^{\mathrm{T}}$，在只变步长的情况下可得

$$z_{ks}=\left(y_k, R_p h\dot{y}_k, \frac{(R_p h)^2}{2!}y_k^{(2)}, \cdots, \frac{(R_p h)^p}{p!}y_k^{(p)}\right)^{\mathrm{T}}$$

$$=\begin{bmatrix}1 & & & \\ & R_p & & \\ & & \ddots & \\ & & & R_p^p\end{bmatrix}z_k$$

当降阶变步长时，例如从 p 阶降为 $p-1$ 阶，步长由 h 变为 $R_{p-1}h$，这时，z_k、z_{ks} 表达式用 z_k^{p-1}，z_{ks}^{p-1} 表示，且易于得到

$$z_k^{p-1}=\left(y_k, h\dot{y}_k, \frac{h^2}{2!}y_k^{(2)}, \cdots, \frac{h^{p-1}}{(p-1)!}y_k^{(p-1)}\right)^{\mathrm{T}}=(z_k, z_k^1, \cdots, z_k^{p-1})^{\mathrm{T}}$$

$$z_{ks}^{p-1}=\left(y_k, R_{p-1}h\dot{y}_k, \frac{(R_{p-1}h)^2}{2!}y_k^{(2)}, \cdots, \frac{(R_{p-1}h)^p}{(p-1)!}y_k^{(p-1)}\right)^{\mathrm{T}}$$

$$=\begin{bmatrix}1 & & & \\ & R_{p-1} & & \\ & & \ddots & \\ & & & R_{p-1}^{p-1}\end{bmatrix}z_k^{p-1}$$

然而，在升阶变步长时，例如，当阶次由 p 阶升到 $p+1$ 阶，步长由 h 变为 $R_{p+1}h$，则这时，z_k、z_{ks} 表达式用 z_k^{p+1}，z_{ks}^{p+1} 表示如下

$$z_k^{p+1}=\left(y_k,h\dot{y}_k,\frac{h^2}{2!}y_k^{(2)},\cdots,\frac{(R_ph)^p}{p!}y_k^{(p)}\ \frac{h^{p+1}}{(p+1)!}y_k^{(p+1)}\right)^{\mathrm{T}}=(z_k,z_k^1,\cdots,z_k^p,z_k^{p+1})^{\mathrm{T}}$$

$$z_{ks}^{p+1}=\left(y_k,R_{p+1}h\dot{y}_k,\frac{(R_{p+1}h)^2}{2!}y_k^{(2)},\cdots,\frac{(R_{p+1}h)^p}{p!}y_k^{(p)},\frac{(R_{p+1}h)^p}{(p+1)!}y_k^{(p+1)}\right)^{\mathrm{T}}$$

但是，$y_k^{(p+1)}$ 是未知的，为此，必须由已知的数据来估计。若采用差分法，即

$$y_k^{(p+1)}\approx\frac{y_k^{(p)}-y_{k-1}^{(p)}}{h}$$

则得到 z_{ks}^{p+1} 的表达式如下

$$z_{ks}^{p+1}=\left(y_k,R_{p+1}h\dot{y}_k,\frac{(R_{p+1}h)^2}{2!}y_k^{(2)},\cdots,\frac{(R_{p+1}h)^p}{(p+1)!}y_k^{(p+1)}\right)^{\mathrm{T}}$$

$$=\begin{bmatrix}1&&&\\&R_{p+1}&&\\&&\ddots&\\&&&R_{p+1}^{p+1}\end{bmatrix}z_k^{p+1}$$

当然，在实际编程时，为了能进一步提高仿真的效率，还会有一些更细致的考虑，比如，步长或阶次调整的目的是提高仿真的精度和仿真速度，但却引起计算量增加。必须寻找合理的方案，以避免过于频繁地变阶或变步长。

(3) 加速收敛问题

从前面的讨论我们可以看到，Gear 法用于病态系统进行仿真时，必须先用显式公式启动，然后用隐式法进行校正，经过多次迭代，达到适当精度后，再往前推进。显然，迭代的收敛速度极大地影响着仿真效能。

下面，让我们仍以三阶 Gear 法为例来分析其迭代收敛的速度，并讨论改进的方法。三阶 Gear 法的隐式公式如下

$$y_{k+1}=\frac{1}{11}(18y_k-9y_{k-1}+2y_{k-2}+6hf_{k+1}) \tag{3.112}$$

其中 f_{k+1} 就是 $f(y_{k+1})$ 的简写，它一般是 y_{k+1} 的非线性函数，式(3.112)是一非线性方程。在迭代计算时，我们采用的迭代公式为

$$\begin{aligned}y_{k+1}^{(i+1)}&=y_{k+1}^{(i)}+\frac{6}{11}h(f_{k+1}^{(i)}-f_{k+1}^{(i-1)})\\&=y_{k+1}^{(i)}+l_0h(f_{k+1}^{(i)}-f_{k+1}^{(i-1)})\end{aligned} \tag{3.113}$$

其中 l_0 是向量 $\boldsymbol{L}$ 的第一个分量(参见式(3.101))，这称为 Picard 迭代。对该迭代过程

$$\begin{cases}\varepsilon_{k+1}^{(i+1)}=y_{k+1}^{(i+1)}-y_{k+1}^{(i)}=l_0h\dfrac{\partial f}{\partial y}\varepsilon_{k+1}^{(i)}\\ \varepsilon_{k+1}^{(i)}=\left(l_0h\dfrac{\partial f}{\partial y}\right)^{(i)}\varepsilon_{k+1}^{(0)}(i=1,2,\cdots)\end{cases} \tag{3.114}$$

为保证收敛，则要求

$$\left|l_0h\frac{\partial f}{\partial y}\right|<1$$

即

$$h\left|\frac{\partial f}{\partial y}\right|<\frac{1}{l_0}$$

一般 $1/l_0$ 大约为 1～3 之间，可见，h 不能太大，否则迭代将不收敛或收敛速度十分缓慢。这

大大限制了 Gear 法的有效性。

为了加速迭代的收敛性，Gear 建议采用牛顿迭代法。实际上，式(3.112)可改写为

$$\begin{aligned} y_{k+1}^{(i+1)} &= l_0 h f(y_{k+1}^{(i)}) + \frac{1}{11}(18y_k - 9y_{k-1} + 2y_{k-2}) \\ &= l_0 h f(y_{k+1}^{(i)}) + c \end{aligned} \tag{3.115}$$

为书写方便，将式(3.115)简记为

$$y = l_0 h f(y) + c \tag{3.116}$$

令

$$g(y) = -y + l_0 h f(y) + c \tag{3.117}$$

显然，式(3.116)与 $g(y)=0$ 同解。由牛顿迭代法，不难得到

$$y^{(i+1)} = y^{(i)} - \frac{g(y^{(i)})}{\left(\frac{\partial g}{\partial y}\right)_{y=y^{(i)}}} \tag{3.118}$$

从式(3.117)求得$\frac{\partial g}{\partial y}$的表达式，并将 $y=y^{(i)}$ 代入$\frac{\partial g}{\partial y}$及式(3.118)，可得

$$y^{(i+1)} = y^{(i)} + \frac{-y^{(i)} + l_0 h f(y^{(i)}) + c}{1 - l_0 h\left(\frac{\partial f}{\partial y}\right)_{y=y^{(i)}}} \tag{3.119}$$

将 $y=y^{(i)}$ 代入式(3.116)，所得结果再代入式(3.119)，可得

$$\begin{aligned} y^{(i+1)} &= y^{(i)} + \frac{-(l_0 h f(y^{(i-1)}) + c) + l_0 h f(y^{(i)}) + c}{1 - l_0 h\left(\frac{\partial f}{\partial y}\right)_{y=y^{(i)}}} \\ &= y^{(i)} + \frac{l_0[h f(y^{(i)}) - h f(y^{(i-1)})]}{1 - l_0 h\left(\frac{\partial f}{\partial y}\right)_{y=y^{(i)}}} \\ &= y^{(i)} + \frac{1}{1 - l_0 h\left(\frac{\partial f}{\partial y}\right)_{y=y^{(i)}}} l_0 \boldsymbol{G}^{(i)} \end{aligned} \tag{3.120}$$

将式(3.120)与式(3.113)进行比较后不难发现，前者多了一个“加速”因子

$$\frac{1}{1 - l_0 h\left(\frac{\partial f}{\partial y}\right)_{y=y^{(i)}}}$$

对微分方程组来说，$\boldsymbol{y}$ 是 N 维向量，用 $\boldsymbol{y}=(y_1, y_2, \cdots, y_N)^{\mathrm{T}}$ 表示，$\boldsymbol{f}$ 是 N 维向量，用 $\boldsymbol{f}=(f_1, f_2, \cdots, f_N)^{\mathrm{T}}$ 表示，式(3.120)成为如下形式

$$\boldsymbol{y}^{(i+1)} = \boldsymbol{y}^{(i)} + \left[\boldsymbol{I} - l_0 h\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right)_{\boldsymbol{y}=\boldsymbol{y}^{(i)}}\right]^{-1} l_0 \boldsymbol{g}^{(i)} \tag{3.121}$$

其中 $\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{g}$ 为 N 维向量，$\boldsymbol{I}$ 为 N 阶单位阵，而$\partial \boldsymbol{f}/\partial \boldsymbol{y}$ 为 $N \times N$ 的方阵，亦称为雅可比矩阵

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}} = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} & \cdots & \frac{\partial f_1}{\partial y_N} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} & \cdots & \frac{\partial f_2}{\partial y_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial y_1} & \frac{\partial f_N}{\partial y_2} & \cdots & \frac{\partial f_N}{\partial y_N} \end{bmatrix} \tag{3.122}$$

这样采用牛顿法，式(3.101)成为

$$\begin{cases} \boldsymbol{z}_{k+1}^{(0)} = \boldsymbol{P}\boldsymbol{z}_k \\ \boldsymbol{z}_{k+1}^{(i+1)} = \boldsymbol{z}_{k+1}^{(i)} + \boldsymbol{L}\left[\boldsymbol{I} - l_0 h\left(\dfrac{\partial f}{\partial y_k}\right)_i\right]^{-1} G_{k+1}^{(i)} \end{cases} \tag{3.123}$$

其中 $\boldsymbol{z}$ 为$(K+1)\times 1$ 的向量，而 G 为标量。对于 y 为 N 维的情况，则有

$$\begin{cases} \boldsymbol{Z}_{k+1}^{(0)} = \boldsymbol{P}\boldsymbol{Z}_k \\ \boldsymbol{Z}_{k+1}^{(i+1)} = \boldsymbol{Z}_{k+1}^{(i)} + \boldsymbol{L}\left[\boldsymbol{I} - l_0 h\left(\dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right)_i\right]^{-1} \cdot (\boldsymbol{g}(\boldsymbol{Z}_{k+1}^{(i)}))^{\mathrm{T}} \end{cases} \tag{3.124}$$

式(3.124)中的 $\boldsymbol{Z}$ 为$(K+1)\times N$ 的矩阵，而 $\boldsymbol{g}$ 为 $N\times 1$ 的向量。

牛顿法能加快迭代过程的收敛，这是该方法的优点，然而，因每次迭代时都必须计算雅可比矩阵，还要计算矩阵求逆，计算量将增加较多。

3.5.6 半隐式龙格-库塔法

在前面几节中我们已经对龙格-库塔法进行了详细讨论，可以看到，对于显式的龙格-库塔法，由于其稳定域的限制，难以用于病态系统的仿真。

实际上，龙格-库塔公式也有隐式表达形式。由于各种龙格-库塔公式可以写成如下形式

$$y_{k+1} = y_k + h\sum_{i=1}^{s} c_i K_i \tag{3.125}$$

其中

$$K_i = f\left(t_k + a_i h, y_k + h\sum_{j=1}^{s} b_{ij} K_j\right) \quad (i = 1,2,\cdots,s) \tag{3.126}$$

h 为仿真时间 t_k 到 t_{k+1}之间的步长，即 $h = t_{k+1} - t_k$；s 为计算右端函数 f 的次数，也称为龙格-库塔法的级数；如果 $j > i-1$ 时 $b_{ij} = 0$，则称为显式龙格-库塔法；然而，如果 $j > i-1$ 时 $b_{ij} \neq 0$，则要求解 s 个联立方程才能计算出 $K_1, K_2, \cdots, K_s$，则称为隐式龙格-库塔法，其中，如果 $j > i$ 时，$b_{ij} = 0$，则称为半隐式龙格-库塔法。

罗森布洛克(Rosenbrock)对半隐式龙格-库塔法作了改进，他在公式中引进了 $\mathrm{d}f(y,t)/\mathrm{d}t = f'(y_k)$，其一般形式为

$$y_{k+1} = y_k + h\sum_{i=1}^{s} c_i K_i \tag{3.127}$$

$$K_i = f\left(t_k + a_i h, y_k + h\sum_{j=1}^{i-1} b_{ij} K_j\right) + h\sum_{j=1}^{i} r_{ij} f'(y_k) K_j \quad (i = 1,2,\cdots,s) \tag{3.128}$$

令 $r_{ii} = \gamma$，则可得

$$[1 - \gamma h f'(y_k)]K_i = f(t_k + a_i h, y_k + h\sum_{j=1}^{i-1} b_{ij} K_j) + h\sum_{j=1}^{i-1} r_{ij} f'(y_k) K_j$$

令

$$[1 - \gamma h f'(y_k)] = B(y_k)$$

则

$$K_i = B^{-1}(y_k)\left[f\left(t_k + a_i h, y_k + h\sum_{j=1}^{i-1} b_{ij} K_j\right) + h\sum_{j=1}^{i-1} \gamma_{ij} f'(y_k) K_j \right] \tag{3.129}$$

例如，典型的三阶三级半隐式龙格-库塔法 Rosenbrock 公式如下

$$y_{k+1} = y_k + \frac{1}{2}h(K_2 + K_3)$$

$$K_1 = B(y_k)^{-1} f(y_k)$$

$$K_2 = B(y_k)^{-1} f(y_k + hb_{21} K_1)$$

$$K_3 = B(y_k)^{-1} f(y_k + hb_{31} K_1 + hb_{32} K_2)$$

其中

$$b_{21} = -0.509\,643\,682\,4, \quad b_{31} = 0.327\,025\,866\,1, \quad b_{32} = 0.310\,884\,773\,1$$

$$\gamma = 0.438\,665\,216$$

显然，在使用半隐式龙格-库塔法仿真时，要求能准确地计算出雅可比矩阵，这是这种方法最基本的要求。另外，这种方法无直接估计计算误差的公式，这给误差控制带来困难。为此，可采用如下处理方法：先用 h 计算一次得到 y_{k+1}，再用 $\frac{1}{2}h$ 计算两次得到 y'_{k+1}，然后令截断误差为

$$E_{k+1} = \frac{|y'_{k+1} - y_{k+1}|}{2^k - 1}$$

其中 k 为公式的阶次。

3.5.7 多帧速算法

实际的病态系统往往可以根据其物理结构而分成几部分(例如分成两部分)，其中一些部分变化快，而另一些部分变化慢。这样，我们可以分别对它们采用不同的步长(甚至不同的算法)计算，即变化快的部分采用小步长(或高精度算法)，变化慢的部分采用较大的步长(或精度较低的算法)，称为多帧速算法。常用的是双帧速算法，即把系统分成两部分而分别计算，其实现思想可以推广到其它多帧速算法。

1. 基于插值的双帧速算法

假设系统已分成快子系统和慢子系统两部分，其快子系统的模型为式(3.130)，而慢子系统如式(3.131)描述，即

$$\dot{\boldsymbol{x}} = \boldsymbol{g}(t, \boldsymbol{x}, \boldsymbol{y}), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{3.130}$$

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{x}), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0 \tag{3.131}$$

其中 $\boldsymbol{x}$ 是快变化部分的状态变量，设为 n 维；$\boldsymbol{y}$ 是慢变化部分的状态变量，设为 l 维。

双帧速算法的基础是恰当地选择算法。这里介绍一种基于变步长四阶龙格-库塔-默尔森法(Runge-Kutta-Merson Method)的双帧速算法。

设系统方程为 $\dot{z} = f(t, z)$，$z = [\boldsymbol{x}, \boldsymbol{y}]^{\mathrm{T}}$，该方法的计算公式如下

$$z_{k+1} = z_k + \frac{h}{6}(K_1 + 4K_4 + K_5) \tag{3.132}$$

$$\begin{cases} K_1 = f(t_k, z_k) \\ K_2 = f\left(t_k + \dfrac{h}{3}, z_k + \dfrac{h}{3}K_1\right) \\ K_3 = f\left(t_k + \dfrac{h}{3}, z_k + \dfrac{h}{6}(K_1 + K_2)\right) \\ K_4 = f\left(t_k + \dfrac{h}{2}, z_k + \dfrac{h}{8}(K_1 + 3K_3)\right) \\ K_5 = f\left(t_k + h, z_k + \dfrac{h}{2}(K_1 - 3K_3 + 4K_4)\right) \end{cases} \tag{3.133}$$

基于插值的双帧速算法的基本思想是在一个积分步长内忽略快、慢子系统之间的耦合，两者之间的数据交换采用插值计算方法。考虑快、慢两部分采用同一方法，仅仅步长不同。下面讨论基于龙格-库塔-默尔森法(简记为 RKM 法)的双帧速算法的实现。

首先根据慢子系统仿真精度要求确定仿真步长 h，在 h 之内两子系统之间的耦合予以忽略。

① 计算慢子系统的 K_1

从 t_k 开始，步长为 h，$K_1 = F(t_k, Y_k, X_k)$

考虑 RKM 法的 K_2 是以 $h/3$ 进行计算，需要对慢子系统的 $Y_{k+1/3}$ 进行插值，得到在 $t_k + h/3$ 处的 $Y_{k+1/3}$ 预估值 $Y'_{k+1/3}$。采用欧拉法，可得

$$Y'_{k+1/3} = Y_k + \frac{h}{3}K_1$$

快子系统在 $t_k \to t_k + \dfrac{h}{3}$ 区间积分，初值为 X_k, Y_k，区间内所需要的 $Y(t)$ 采用线性插值的办法

$$Y(t) = Y_k + (t - t_k)K_1 \quad \left(t_k \leqslant t \leqslant t_k + \frac{h}{3}\right)$$

其步长 h' 由仿真精度决定，计算得到 $X_{k+1/3}$。

② 计算慢子系统的 K_2

$$K_2 = F(t_{k+1/3}, Y'_{k+1/3}, X_{k+1/3})$$

这时，对 $Y'_{k+1/3}$ 用改进欧拉法进行校正

$$Y''_{k+1/3} = Y_k + \frac{1}{2} \cdot \frac{h}{3}(K_1 + K_2)$$

如果需要，也可以利用它再对快子系统进行积分，以校正 $X_{k+1/3}$，校正后的 $X_{k+1/3}$ 可进一步用于改进 K_2 及 $Y_{k+1/3}$，通过迭代计算，提高计算精度。

③ 计算慢子系统的 K_3

$$K_3 = F(t_{k+1/3}, Y_{k+1/3}, X_{k+1/3})$$

注意，这里采用 $Y_{k+1/3}$ 而不是 $Y'_{k+1/3}$ 或 $Y''_{k+1/3}$，是指迭代后得到的慢子系统状态变量的值。然后计算 $t_{k+1/2}$ 处的 $Y_{k+1/2}$ 估计值

$$Y_{k+1/2} = Y_k + \frac{h}{8}(K_1 + K_3)$$

同样，也要计算快子系统的 $X_{k+1/2}$，为此，对 $Y(t)$ 在 $\left(t_k \leqslant t \leqslant t_k + \dfrac{h}{2}\right)$ 之间的值进行插值

$$\begin{aligned} Y(t) &= Y_{k+1/3} + (t - t_{k+1/3}) \frac{Y_{k+1/2} - Y_{k+1/3}}{\dfrac{h}{2} - \dfrac{h}{3}} \\ &= Y_{k+1/3} - (t - t_{k+1/3})(K_1/4 + K_2 + 9K_3/4) \end{aligned}$$

这样，利用RKM法和上述$Y(t)$可计算得到$X_{k+1/2}$，积分步长按精度要求选择。

④ 计算慢子系统的K_4

$$K_4 = F(t_{k+1/2}, Y_{k+1/2}, X_{k+1/2})$$

再利用下式估计慢子系统的Y_{k+1}的值Y'_{k+1}

$$Y'_{k+1} = Y_k + \frac{h}{2}(K_1 - 3K_2 + 4K_3)$$

对快子系统，进行$(t_{k+1/2}, t_{k+1})$区间的积分。同样，对该区间的$Y(t)$也采用插值的方法

$$Y(t) = Y_{k+1/2} + (t - t_{k+1/2})K_4 \quad (t_{k+1/2} \leqslant t \leqslant t_{k+1})$$

积分后得到X_{K+1}。

⑤ 计算慢子系统的K_5

$$K_5 = F(t_{k+1}, Y'_{k+1}, X_{k+1})$$

同时可计算得到

$$Y_{k+1} = Y_k + \frac{h}{6}(K_1 + 4K_4 + K_5)$$

为了提高精度，第④、⑤两步可进行迭代计算。

⑥ 确定下一步的积分步长

积分步长的确定基于计算误差。RKM法的截断误差有解析表达式如下

$$E_k = h(K_1/3 - 3K_3/2 + 4K_4/3 - K_5/6)$$

其中E_k为本步计算截断误差。根据计算精度的要求，可确定下一步计算步长。

上述各步中没有显式地列出快子系统积分计算过程，读者从前面列出的RKM法的计算公式中不难理解其计算过程。

2. 波形松弛法

这种方法是在整个积分过程中忽略子系统之间的耦合。为了达到所要求的精度，需要进行迭代计算。

若系统方程仍如式(3.130)与式(3.131)所示。从其初值出发，例如先对式(3.130)进行积分，但在积分过程中缺乏$Y(t)$的值。波形松弛法假定$Y(t)$的估计值为$^{[0]}Y$(一般在一开始，可用Y_0作为第一次估计值)，式(3.130)成为下式

$$\frac{dX}{dt} = G(t, X, {}^{[0]}Y), \quad X(0) = X_0$$

然后便可对其进行积分计算，得到第一次逼近$^{[1]}X$。接着，式(3.131)成为下式

$$\frac{dY}{dt} = F(t, Y, {}^{[1]}X), \quad Y(0) = Y_0$$

对其进行积分计算，得到的$^{[1]}Y$作为近似解，再对式(3.130)进行积分，得到更好的逼近值$^{[2]}X$，这种过程一直进行下去，直到两次迭代误差满足规定的精度为止。这称为高斯-赛得尔迭代，其一般表达式如下。首先从下式解$^{[i+1]}X$

$$\frac{d^{[i+1]}X}{dt} = G(t, {}^{[i+1]}X, {}^{[i]}Y), \qquad {}^{[i+1]}X(0) = X_0$$

得到$^{[i+1]}X$后从下式解$^{[i+1]}Y$

$$\frac{\mathrm{d}^{[i+1]}Y}{\mathrm{d}t} = F(t,{}^{[i+1]}Y,{}^{[i+1]}X), \qquad {}^{[i+1]}Y(0) = Y_0$$

另一种波形松弛法称为高斯-雅可比迭代法，它与高斯-赛得尔迭代的区别是：设$X(t)$，$Y(t)$的初始估计值为${}^{[0]}X$，${}^{[0]}Y$，然后分别积分下列方程

$$\frac{\mathrm{d}X}{\mathrm{d}t} = G(t,X,{}^{[0]}Y), \quad X(0) = X_0$$

$$\frac{\mathrm{d}Y}{\mathrm{d}t} = F(t,Y,{}^{[0]}X), \quad Y(0) = Y_0$$

得到较好的逼近值${}^{[1]}X$，${}^{[1]}Y$，重复此过程，其一般表达式如下

$$\frac{\mathrm{d}^{[i+1]}X}{\mathrm{d}t} = G(t,{}^{[i+1]}X,{}^{[i]}Y), \qquad {}^{[i+1]}X(0) = X_0$$

$$\frac{\mathrm{d}^{[i+1]}Y}{\mathrm{d}t} = F(t,{}^{[i+1]}Y,{}^{[i]}X), \qquad {}^{[i+1]}Y(0) = Y_0$$

同样，这种过程一直进行下去，直到两次迭代误差满足规定的精度为止。

3.6 间断特性的仿真建模*

从前面的讨论中我们知道，当用经典的数值积分法(包括单步法和多步法)进行仿真时，右端函数的连续性对保证仿真精度是十分重要的。单步法要求在一个步长内右端函数连续，K 阶多步法则要求在 K 步内右端函数连续。为了使仿真结果达到一定精度，人们往往采用变步长方法。然而，采用变步长方法也会遇到困难，下面，我们对此稍加讨论。

若函数的形式如图 3.12 所示。设当前的仿真时刻为t_k，仿真步长为 h_k。下一步，首先按 h_k 计算，这时，可能跨过间断点 t^*，显然，计算误差超过要求，为此，减少步长 h_k，即从 t_{k+1}退回到 $t_{k+1}^{(1)}$，这时，只有当 $t_{k+1}^{(1)} < t^*$ 时误差才可能达到要求。然而此时，由于 $f(y,t)$处于连续，步长小，误差也很小，又进入放大步长状态，结果又跨过 t^* 点。这样，在间断点，步长的调整要反复多次，才能找到间断点，从而计算量大大增加。

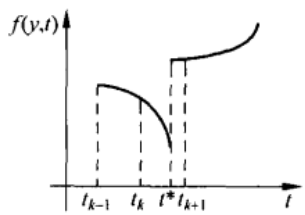


图 3.12 间断特性仿真过程示意

为了增加读者的感性认识，下面给出一个实际例子。设系统可用如下微分方程描述

$$\begin{cases} \dfrac{\mathrm{d}y_1}{\mathrm{d}t} = y_2 & y_1(0) = -0.45 \\ \dfrac{\mathrm{d}y_2}{\mathrm{d}t} = \begin{cases} 0.5 & -0.5y_2 \geqslant y_1 \\ -0.5 & -0.5y_2 \leqslant y_1 \end{cases} & y_2(0) = 0.975 \end{cases} \tag{3.134}$$

现要求计算 $t=0\sim4.0$ 这一段的过渡过程，采用 RKM-34 法进行仿真，相对误差限为 $10^{-4}\sim10^{-6}$。步长控制策略为：当实际相对误差大于10^{-4}时，步长缩小一半，若当实际相对误差小于10^{-6}时，则步长放大一倍，初始步长取 0.1。表 3.10 给出其中一个间断点的仿真计算过程。

表 3.10 式(3.134)系统仿真过程

t	y_1	y_2	h	最大相对误差
2.8	0.319 99	−0.425 000 2	0.1	0.730 449E-10
2.9	0.274 997	−0.475 000 2	0.1	0.225 775 2E-9
3.0	×	×	0.1	0.216 216 2E-2
2.95	√	√	0.05	0.311 658 4E-9
3.0	×	×	0.05	0.222 222E-2
2.975	×	×	0.025	0.111 111E-2
2.962 5	×	×	0.012 5	0.555 55E-3
2.956 25	×	×	0.006 25	0.277 778E-3
2.953 125	×	×	0.003 125	0.555 55E-3
2.951 562 5	√	√	0.001 562 5	0.346 98E-4
2.953 125	√	√	0.001 562 5	0.173 868E-10
2.956 25	√	√	0.003 125	0.248 539E-11
2.962 5	√	√	0.006 25	0.497 7E-11
2.975	√	√	0.012 5	0.399 15E-10
3.0	0.226 185	−0.476 302 3	0.025	0.120 34E-9
3.05	√	√	0.05	0.405 081 8E-10
3.10	0.181 055 2	−0.426 302 3	0.05	0.165 156 2E-9
3.20	0.140 924 9	−0.376 202 3	0.1	0.544 672 9E-9

从表 3.10 中不难看出，在 $t=2.9\sim3.0$ 之间有一个间断点，在初始步长为 0.1 的情况下，为越过该间断点，变步长的 RKM-34 进行了 6 次变步长失败才找到间断点，然后又经过 6 次变步长才重新按恰当步长计算。

因而，如何快速地搜索出间断点，以便能快速地越过间断点是这类系统仿真中需要解决的突出问题。本节将介绍三种基本方法，就是条件函数零点搜索法、平均值法和间断点估计法。

3.6.1 条件函数零点搜索法

本小节讨论系统具有一般间断特性时的仿真建模问题。若系统微分方程 $\dot{y}=f(y,t)$ 中右端函数 $f(y,t)$ 不连续，不失一般性，可用以下形式来描述

$$\frac{\mathrm{d}y}{\mathrm{d}t}=\begin{cases} f_1(y,t), & \phi_1(y,t)\geqslant 0 \\ f_2(y,t), & \phi_2(y,t)\geqslant 0 \\ \vdots & \vdots \\ f_m(y,t), & \phi_m(y,t)\geqslant 0 \end{cases} \tag{3.135}$$

即右端函数的表达式依赖于条件函数的取值。

对式(3.135)所描述的系统(为了简单起见，我们仅考虑单变量系统，对多变量系统，可用矩阵形式描述)，若能事先确定 $f(y,t)$ 满足条件函数的点，即满足 $\phi_i(y,t)=0$ 的 t^*，然后就可以恰当地选择步长。

根据条件函数的特点，可以将其分为两大类：一类是条件函数仅与时间有关，即

$\phi_i(y,t)=\phi(t)=0$；一类是一般情况，即不仅与时间有关，而且同时与变量的值有关。

对第一种情况，仿真推进比较简单。事先确定满足条件函数的时间点值，仿真钟推进时，每一步均与间断点时间值进行比较，使当前步不跨过间断点。至于 $\phi(t)=0$ 求解即多项式求根的方法很多，这里就不作介绍了。

下面我们重点讨论一般情况，此时条件函数不仅与时间有关，而且与系统变量的值有关。此时，条件函数的零点无法事先求出，而必须在每一步仿真计算时进行判断。这里的主要困难在于仿真步长选择多大为宜以免跨过间断点。

若从某一时刻 t_k 出发，当前时刻的系统变量值 y_k 已经得到，现在仿真钟按某一步长（设为 h_k）往前推进到 $t_{k+1}=t_k+h_k$，这时得到的系统变量预报值 $y_{k+1}^{(1)}$，相应地可以计算得到 $\phi_k(y_k,t_k)$，$\phi_{k+1}(y_{k+1},t_{k+1})$，同时也可计算得到 $\dot{\phi}_k(y_k,t_k)$，$\dot{\phi}_{k+1}(y_{k+1},t_{k+1})$。用这些值构造一个插值函数 $\phi(t)$，用它来取代原来的条件函数。由于此时的条件函数仅与时间有关，因而易于求得它的根，设为 $t_{k+1}^*=t_k+\Delta t_k$，也就是间断点的时间值，然后再调整步长，从 (t_k,y_k) 出发，按步长 $h_{k+1}=\Delta t_k$ 推进到 t_{k+1}^*，这样就不会跨过间断点。

这种方法的关键是要找到一种合适的插值函数。这里介绍一种使用比较普遍的插值函数，即 Hermiter 内插公式，其形式如下

$$\begin{aligned}\phi(t)=&[3(1-a)^2-2(1-a)^3]\phi_k(y_k,t_k)+[3a^2-2a^3]\phi_{k+1}(y_{k+1}^{(1)},t_{k+1})\\&+h_k[(1-a)^2-(1-a)^3]\dot{\phi}_k(y_k,t_k)-h_k[a^2-a^3]\dot{\phi}_{k+1}(y_{k+1}^{(1)},t_{k+1})=0\end{aligned}\tag{3.136}$$

其中 $a=(t-t_k)/h_k$。

当然，由于 ϕ_{k+1}，$\dot{\phi}_{k+1}$ 均是基于预报值 $y_{k+1}^{(1)}$ 计算出来的，而 $y_{k+1}^{(1)}$ 是基于 $f(y,t)$ 连续导出的，因而难免会有误差，甚至有可能产生失误。但是，只要 h_k 不是特别大，这种方法仍然是可用的。下面，我们用一个实例来说明这种方法的有效性。系统的微分方程如式(3.137)

$$\frac{dy}{dt}=\begin{cases}-y+\sin t, & |y|<0.5\\-0.5y+\sin t, & y\geqslant 0.5\\-0.2y+\sin t, & y\leqslant -0.5\end{cases}\qquad\begin{matrix}t_0=\pi/4\\y(t_0)=0\end{matrix}\tag{3.137}$$

首先对其条件函数进行改造。不难发现，可令

$$\begin{cases}\phi_1=y-0.5\\\phi_2=-y-0.5\end{cases}$$

则式(3.137)可改写成

$$\frac{dy}{dt}=\begin{cases}-y+\sin t, & \begin{cases}\phi_1<0\\\phi_2<0\end{cases}\\-0.5y+\sin t, & \phi_1\geqslant 0\\-0.2y+\sin t, & \phi_2\geqslant 0\end{cases},\quad\begin{cases}t_0=\pi/4\\y(t_0)=0\end{cases}\tag{3.138}$$

仿真结果表明，在 $t=\pi/4\sim 12.5$，条件函数共有 7 个零点，若要求计算精度为 $\varepsilon=10^{-5}$，采用变步长 RKF-45 法计算右端函数 $f(y,t)$ 的次数为 1136 次，而采用一般四阶 RK 法及 Hermite 插值仅计算 360 次。

总之，条件函数零点搜索法是解决间断点仿真的有效方法，其特点是必须将间断点的条

件转化为显式条件函数。当然，多项式求解算法需花费一些时间，但从总体上来讲，这种方法的效果是显著的。

3.6.2 平均值法

条件函数零点搜索法通过准确地搜索其零点以确定仿真步长从而保证间断点的准确计算。当然，这是以搜索零点的计算为代价的。在高精度仿真时，这种方法得到了广泛的应用。然而，比如说，在采用离散相似法进行仿真时，如果作用函数的重构器采用零阶或一阶，而未加其他补偿器，其计算精度有限，在这种情况下，对间断点的高精度搜索有时效果并不显著，相反，却大大影响了仿真速度。于是，人们力图找到一种精度适中、计算速度较快的间断点仿真建模方法，以满足仿真速度的要求。平均值法就是一种典型的方法。

1. 基本思想

设间断特性满足如下关系

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(y,u) \tag{3.139}$$

其中 $f(y,u)$ 中包含间断点，u 为作用函数，y 为被仿真的系统变量。设在第 k 步到第 $k+1$ 步作用函数 u 存在间断点，则由式(3.139)

$$\begin{aligned} y_{k+1} &= y_k + \int_{kh}^{(k+1)h} f(y,u)\mathrm{d}t \\ &= y_k + \int_{u_k}^{u_{k+1}} \frac{f(y,u)}{\frac{\mathrm{d}u}{\mathrm{d}t}}\mathrm{d}u = y_k + \int_{u_k}^{u_{k+1}} \frac{f(y,u)}{g(y,u)}\mathrm{d}u \end{aligned} \tag{3.140}$$

式中，h 表示步长。平均值法的做法是，在区间$[kh,(k+1)h]$，$\mathrm{d}u/\mathrm{d}t$=常数，即

$$g(y,u) = \frac{u_{k+1}-u_k}{h} = \frac{\Delta u_k}{h} \tag{3.141}$$

从而

$$y_{k+1} = y_k + h \cdot \frac{1}{\Delta u_k}\int_{u_k}^{u_{k+1}} f(y,u)\mathrm{d}u$$

记

$$f_{\mathrm{ave}}(u_k,\Delta u_k) = \frac{1}{\Delta u_k}\int_{u_k}^{u_{k+1}} f(y,u)\mathrm{d}u \tag{3.142}$$

称为 $f(y,u)$ 的平均函数，则

$$y_{k+1} = y_k + h \cdot f_{\mathrm{ave}}(u_k,\Delta u_k) \tag{3.143}$$

2. 典型间断特性的平均函数

由上面的讨论可以看到，平均值法的关键是如何得到间断特性的平均函数。本节我们导出几种典型间断特性的平均函数，读者可以将该导出过程推广到一般情况。

(1) 继电特性

典型的继电特性如图 3.13 所示，显然可令 $f(y,u)=u/|u|(u\neq 0)$，这样，由式(3.142)我们可以得到

$$f_{ave}(u_k,\Delta u_k)=\frac{1}{\Delta u_k}\int_{u_k}^{u_{k+1}}f(y,u)\mathrm{d}u=\frac{1}{\Delta u_k}\int_{u_k}^{u_{k+1}}\frac{u}{|u|}\mathrm{d}u$$
$$=\frac{1}{\Delta u_k}\int_{u_k}^{u_k+\Delta u_k}\frac{u}{|u|}\mathrm{d}u=\frac{1}{\Delta u_k}(|u|)\Big|_{u_k}^{u_k+\Delta u_k}$$
$$=\frac{|u_k+\Delta u_k|-|u_k|}{\Delta u_k} \tag{3.144}$$

下面进一步讨论一般情形继电特性的平均函数的计算。此时的继电特性可用图 3.14 来描述。

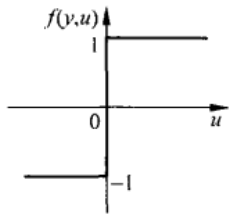


图 3.13 典型继电特性

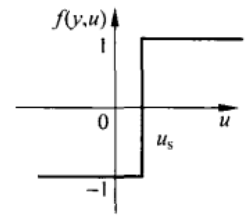


图 3.14 一般继电特性

可以用 $u-u_s$ 代替典型继电特性表达式中的 u，即

$$f(y,u)=(u-u_s)/|u-u_s|\ (u\neq u_s)$$

这样，可得到

$$f_{ave}(u_k,\Delta u_k)=\frac{1}{\Delta u_k}\int_{u_k}^{u_{k+1}}\frac{u-u_s}{|u-u_s|}\mathrm{d}u$$
$$=\frac{1}{\Delta u_k}\int_{u_k}^{u_k+\Delta u_k}\frac{u-u_s}{|u-u_s|}\mathrm{d}u$$
$$=\frac{1}{u_k-u_s}|u-u_s|\Big|_{u_k}^{u_k+\Delta u_k}$$
$$=\frac{|u_k-u_s+\Delta u_k|-|u_k-u_s|}{u_k-u_s} \tag{3.145}$$

(2) 失灵区特性

典型的失灵区特性可用图 3.15 表示，显然，不难得到

$$f(y,u)=(u-u_s+|u-u_s|)/2+(u+u_s+|u+u_s|)/2$$

这样，我们可以求得

$$f_{ave}(u_k,\Delta u_k)=\frac{1}{\Delta u_k}\int_{u_k}^{u_k+\Delta u_k}[(u-u_s+|u-u_s|)/2+(u+u_s+|u+u_s|)/2]\mathrm{d}u$$
$$=\frac{1}{2\Delta u_k}\left\{\frac{1}{2}[(u-u_s)^2+(u-u_s)|u-u_s|]\right.$$
$$\left.+\frac{1}{2}[(u+u_s)^2-(u+u_s)|u+u_s|]\right\}\Big|_{u_k}^{u_k+\Delta u_k}$$
$$=\frac{1}{4\Delta u_k}\{[(u_k+\Delta u_k-u_s)^2-(u_k-u_s)^2$$
$$+(u_k+\Delta u_k-u_s)|(u_k+\Delta u_k-u_s)|-(u_k-u_s)|u_k-u_s|]$$

$$+[(u_k+\Delta u_k+u_s)^2-(u_k+u_s)^2 +(u_k+\Delta u_k+u_s)\mid(u_k+\Delta u_k+u_s)\mid-(u_k+u_s)\mid u_k+u_s\mid]\} \tag{3.146}$$

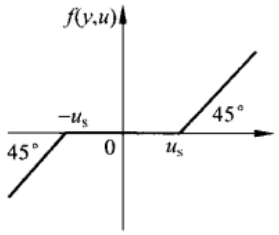


图 3.15 典型失灵区特性

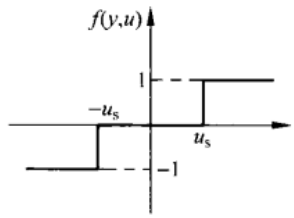


图 3.16 带失灵区的继电特性

(3) 带失灵区的继电特性

这类间断特性既具有失灵区的特点,也具有继电特性的特点,可用图 3.16 来描述,相应地,其函数描述式如下

$$f(y,u)=\frac{1}{2}\frac{u-u_s}{\mid u-u_s\mid}+\frac{1}{2}\frac{u+u_s}{\mid u+u_s\mid}$$

因此,我们可以得到

$$f_{ave}(u_k,\Delta u_k)=\frac{1}{\Delta u_k}\int_{u_k}^{u_k+\Delta u_k}f(y,u)\mathrm{d}u =\frac{1}{2\Delta u_k}[\mid u_k-u_s+\Delta u_k\mid-\mid u_k-u_s\mid+\mid u_k+u_s+\Delta u_k\mid-\mid u_k+u_s\mid] \tag{3.147}$$

(4) 饱和非线性特性

典型饱和非线性特性可用图 3.17 表示。用函数描述则为

$$f(y,u)=\frac{\mid u+u_s\mid-\mid u-u_s\mid}{2}$$

按照前面同样的方法可以得到其平均函数如下

$$\begin{aligned}f_{ave}(u_k,\Delta u_k)&=\frac{1}{\Delta u_k}\int_{u_k}^{u_k+\Delta u_k}f(y,u)\mathrm{d}u\\&=\frac{1}{4\Delta u_k}[\mid u+u_s\mid(u+u_s)-\mid u-u_s\mid(u-u_s)]\Big|_{u_k}^{u_k+\Delta u_k}\\&=\frac{1}{4\Delta u_k}[\mid u_k+\Delta u_k+u_s\mid(u_k+\Delta u_k+u_s)-\mid u_k+u_s\mid(u_k+u_s)\\&\quad-\mid u_k+\Delta u_k-u_s\mid(u_k+\Delta u_k-u_s)+\mid u_k-u_s\mid(u_k-u_s)]\end{aligned} \tag{3.148}$$

其他还有一些典型非线性特性,如回环特性等,均可以用上述方法得到其平均函数。

读者可能觉得上述平均函数形式比较啰唆。实际上,只要平均函数得到,从仿真计算来说,其计算量是不大的,不但可以大大提高仿真速度,而且也可得到适当的精度。

下面通过一个简单的例子来说明如何用平均值法建立间断特性仿真模型。系统的结构图如图 3.18 所示。

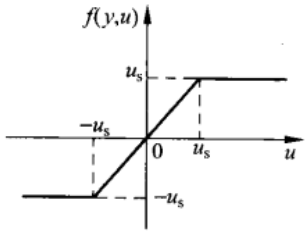


图 3.17 典型饱和非线性特性

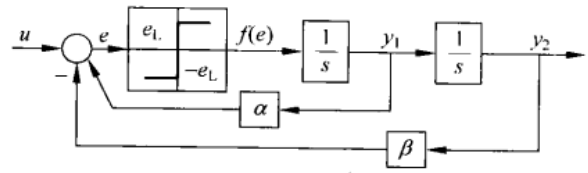


图 3.18 带继电特性的系统

假设我们采用二阶龙格-库塔法对系统进行仿真，间断特性用平均值法建立仿真模型，不难得到

$$e_k = u_k - \alpha y_{1,k} - \beta y_{2,k}$$

$$\hat{y}_{1,k+1} = y_{1,k} + hf(e_k)$$

$$\hat{y}_{2,k+1} = y_{2,k} + h\dot{y}_{2,k} = y_{2,k} + h\hat{y}_{1,k}$$

$$\hat{e}_{k+1} = u_{k+1} - \alpha\hat{y}_{1,k+1} - \beta\hat{y}_{2,k+1}$$

$$\Delta\hat{e}_k = \hat{e}_{k+1} - e_k$$

$$f_{ave}(e_k, \Delta\hat{e}_k) = \frac{|e_k + \Delta\hat{e}_k| - |e_k|}{\Delta\hat{e}_k}$$

$$y_{1,k+1} = y_{1,k} + \frac{h}{2}(\dot{y}_{1,k} + \dot{\hat{y}}_{1,k+1})$$

$$= y_{1,k} + he_L f_{ave}(e_k, \Delta\hat{e}_k)$$

$$y_{2,k+1} = y_{2,k} + \frac{h}{2}(\dot{y}_{2,k} + \dot{\hat{y}}_{2,k+1})$$

$$= y_{2,k} + \frac{h}{2}(y_{1,k} + \hat{y}_{1,k+1})$$

从上述表达式不难看出，平均值法较之一般的龙格-库塔法的计算工作量几乎没有增加。实例表明，其精度却有较大的提高。

例如，设 $u(t)=0, y_1(t_0)=0.975, y_2(t_0)=-0.45, \alpha=0.5, \beta=1.0, e_L=0.5$，选择 $h=0.1$ 进行仿真计算。该系统在 $t=2.9\sim3.0$ 之间作用函数有间断点。分别用一般二阶龙格-库塔法、四阶龙格-库塔法及带平均值法的二阶龙格-库塔法的仿真结果如表 3.11 所示。

表 3.11 仿真结果比较

t	精确解	平均值 RK-2	RK-2	RK-4
2.9	0.275 001	0.275 001	0.275 001	0.275 001
3.0	0.226 25	0.226 226	0.225 001	0.225 382 4
3.1	0.181 25	0.181 126	0.177 501	0.181 055 2

从表 3.11 中可以看到，间断点之前三种方法的计算结果均相同，而间断点以后，带平均值法的二阶龙格-库塔法计算精度最高。

3.6.3 间断点估计法

前面我们讨论了两类间断点仿真建模方法，尽管它们已经包括了许多类型的系统，然而，由于系统的复杂性，总有一些系统的间断特性难以用间断函数描述，甚至难以列出它们的条件函数。因此，人们直接面向仿真算法开展了大量研究，以适用于具有间断点特性的系统进行仿真。这就意味着，这些算法既不搜索条件函数零点，也不计算平均值函数，甚至不需要知道条件函数，而是在仿真过程中自动高效率地估计出系统的间断点，我们将它称为“间断点估计法”。

从前面讨论中我们知道，一般的仿真算法在跨过间断点时要反复搜索多次才能找到间断点，因此，间断点估计法的核心是如何减少搜索次数，使得积分步的末尾正好在间断点上。下面，我们以 Shampine 提出的 RKS-34 公式对为例加以说明。

对微分方程 $\mathrm{d}y/\mathrm{d}t=f(y,t)$，若采用 RKS-34 公式对仿真，其计算表达式为

$$y_{k+1}=y_k+\frac{h}{8}(K_1+3K_2+3K_3+K_4)$$

其中

$$K_1=f(t_k,y_k)$$

$$K_2=f\left(t_k+\frac{h}{3},y_k+\frac{h}{3}K_1\right)$$

$$K_3=f\left(t_k+\frac{2h}{3},y_k+\frac{h}{3}(-k_1+3K_2)\right)$$

$$K_4=f(t_k+h,y_k+h(K_1-K_2+K_3))$$

该公式对的另一个公式是

$$y_{k+1}=y_k+\frac{h}{32}(-K_1+3K_2-3K_3-3K_4+4K_5)$$

$$k_5=f(t_k+h,y_k+\frac{h}{8}(K_1+3K_2+3K_3+K_4))$$

仿真计算时，K_1,K_2,K_3,K_5 分别是 $t_k,t_k+\frac{h}{3},t_k+\frac{2h}{3},t_k+h$ 时间点导函数 $f(t,y)$ 的近似值。该方法的特点是子步长正好将 h 分为三等分。

在仿真过程中，先不考虑间断点的影响，分别计算出 K_1、K_2、K_3、K_4、K_5 以及 $Df_1=K_2-K_1$、$Df_2=K_3-K_2$、$Df_3=K_5-K_3$（参见图 3.19），如果在$[t_k,t_k+h]$区间 $f(t,y)$是连续的，那么有理由认为 Df_1、Df_2、Df_3 的值相差不会很大。反之，如果它们之间的差别很大，比如说某一个远远大于其它两个（10 倍以上），那么，有理由认为该子区间存在间断点。这时，将该子区间再分为三等分，重复进行一次积分，直到它们之间的差别达到要求为止。由于每次将区间缩短到原来的三分之一，因此，很快就可搜索到间断点。值得注意的是，这种方法不需要增加太多的计算量，效率较高。

为了使读者对这种方法的效率有一个感性认识，下面给出一个实例。设系统的方程为 $\mathrm{d}y/\mathrm{d}t=-y+u$，$y(t_0)=0$，$u(t)$如图 3.20 所示。该系统的精确解是：当 $t=4.0$ 时 $y=$

3.939 620 5，表 3.12 中给出了分别用一般 RKS-34 和带间断点估计的 RKS-34 仿真的结果。

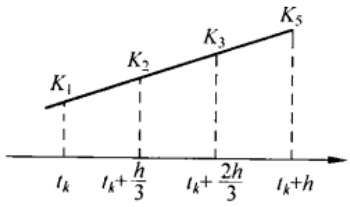


图 3.19 RKS-34 系数计算

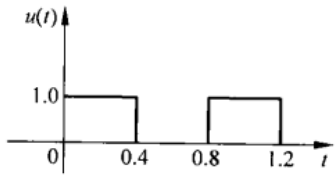


图 3.20 作用函数 $u(t)$

表 3.12 仿真结果比较

误差限	断点搜索次数		$y(t=4.0)$值	
	一般 RKS-34	带间断点估计 RKS-34	一般 RKS-34	带间断点估计 RKS-34
10^{-5}	425	373	3.933 715	3.948 771
10^{-6}	1 805	1 001	3.939 615	3.939 621

由表 3.12 中的数据可以看到，两者的计算精度接近，但断点搜索次数随着精度要求的提高相差很大。实践表明，带间断点的方法可比不带间断点的方法减少搜索次数 30%～50%。

3.7 小结

本章讨论了经典的连续系统仿真建模方法学。被仿真系统的状态变化随时间连续性变化，在数字计算机上对连续系统进行仿真时，首先遇到的问题是，数字计算机的数值及时间均具有离散性，因此离散化原理是数字仿真所必须遵循的。

本章对经典的数值积分法，包括单步法和多步法，从原理到方法进行了系统介绍，读者应该注意到，所有的讨论均从保证算法的稳定性、准确性及快速性这三个方面来进行的，其关键参数是仿真步长。在使用这些方法时，步长的选择必须十分小心，这是离散化原理的要求。

本章还对病态系统的仿真进行了讨论，实际系统，特别是非线性系统，从仿真的角度来看，方法与步长的选择十分重要，仿真过程中必须克服病态现象，这依赖于病态探测与误差估计。

经典的数值积分法的数学基础是要求被积导函数连续，因此，从仿真的角度来看，间断点也是一种病态问题。因此，间断点的处理对高精度的仿真至关重要。

经典的数值积分法已经经过多年的研究，理论和方法相当完备，本书只是择其要点进行讨论，详细内容读者可参考相关书籍和文献。

习题

1. 已知方程$\dot{y}=-\frac{1}{\tau}y$(τ 为系统的时间常数)，试用四阶龙格-库塔法进行积分计算。为保证系统设计稳定，计算步距 h 最大不能超过多少。

2. 试推导二阶龙格-库塔法的截断误差系数。

3. 已知电路如图 3.21 所示，设初始条件为零，电压源是单位阶跃。用改进节点法列出电路方程，然后分别给出欧拉法和梯形法的仿真模型。取步长为 0.1s。

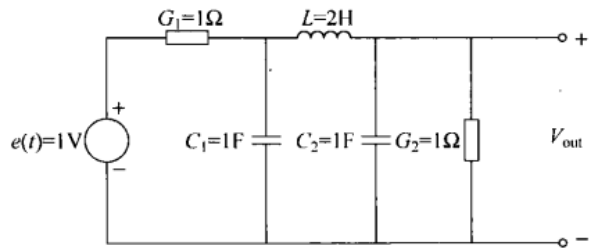


图 3.21　习题 3 电路图

4. 已知微分方程$\dot{y}=e^{-y}-t^2$，分别用二阶、四阶龙格-库塔法列出 $y(t)$的差分方程。

5. 试比较分析单步法与多步法的异同。

6. 试推导由 x_{n+k-1}和$\dot{x}_{n+k-1}$预报 x_{n+k}的公式并计算其截断误差系数。

7. 试推导由 x_{n+k}、x_{n+k-1}和$\dot{x}_{n+k-1}$计算$\dot{x}_{n+k}$的表达式并计算其截断误差系数。

8. 已知线性多步法公式为

$$x_{n+k}=x_{n+k-1}+\frac{h}{12}(23\,\dot{x}_{n+k-1}-16\,\dot{x}_{n+k-2}+5\,\dot{x}_{n+k-3})$$

试用测试方程分析该方法的稳定域。

9. 已知系统用如下微分方程描述

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}u,\quad \boldsymbol{x}=\begin{bmatrix}x_1\\x_2\end{bmatrix},\quad \boldsymbol{x}(t_0=0)=\begin{bmatrix}0\\0\end{bmatrix},\quad u(t)=1(t)$$

$$\boldsymbol{A}=\begin{bmatrix}0 & -100\\1.0 & -100\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}100\\0\end{bmatrix}$$

试用增广矩阵的蛙跳法计算系统的动态响应。

10. 用二阶显式 Gear 法预报，再用二阶 Gear 法校正，对如下方程进行仿真

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}=\begin{bmatrix}-21 & 19 & -20\\19 & -21 & 20\\40 & -40 & 40\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix},\quad \boldsymbol{x}(t_0=0)=\begin{bmatrix}1\\0\\-1\end{bmatrix}$$

11. 将二阶 Gear 法转换为单步多值法，然后对 1 题及 2 题进行仿真，并比较其结果。

12. 试推导三阶阿达姆氏公式的 $\boldsymbol{L}$ 向量为：$\boldsymbol{L}=(l_0,l_1,l_2,l_3)=\left(\frac{5}{12},1,\frac{3}{4},\frac{1}{6}\right)$。

13. 已知系统的结构如图 3.22 所示，试用带平均值法的二阶龙格-库塔法计算系统输出过程。其中 $b=c=1.0, R=1(t)$。先计算非线性环节的平均函数，再列出计算方程，然后计算出输出过程(步长 h 自行确定)。

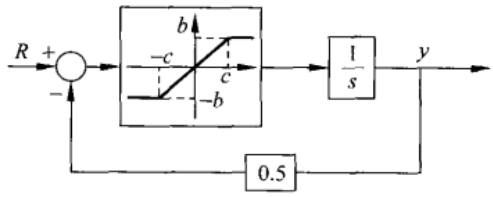


图 3.22 系统结构图

14. 试求下列典型非线性环节的平均函数：

(1) 带失灵区的限幅环节(图 3.23)　　(2) 回环特性(图 3.24)

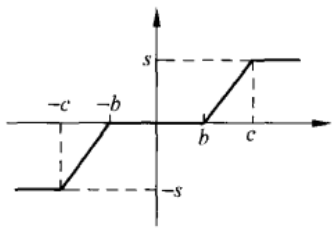


图 3.23 带失灵区的限幅环节($s=c-b$)

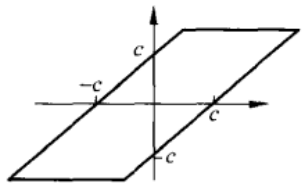


图 3.24 回环特性

参考文献

[1] Gear C W 著. 费景高等译. 常微分方程初值问题的数值解法. 北京：科学出版社，1978

[2] 熊光楞. 数字仿真算法与软件. 北京：宇航出版社，1991

[3] Fehlberg E. Low-Order Classical Runge-Kutta Formulas with Step-Size Control and Their Application to Some Heat Transfer Problems. NASA Report TR R-315 , George C. Malchall Space Flight Creater, April,15,1969

[4] Chai A S. Modified Merson's Integration Algorithm Which Saves two Evaluation at Each Step, Simulation ,March 1974

[5] Shampine L F. Solving Ordinary Differential Equations for Simulation . Math. Computer in Simulation, 1978

[6] Bernstein D S. The Treatment of Inputs in Real Time Digital Simulation. Smulation,1979,33(2)

[7] Howe R M. Computer Consideration for Real Time Simulation. NASA Technical Report, CR-2877,1977

[8] Vlach J 等著. 汪惠等译. 电路分析和设计的计算机方法. 北京：科学出版社，1992

[9] Brayton R K. Error estimates for the variable-step backward differentiation methods. IBM Research Report RC 6205(#26655),September 1976

[10] Brayton R K, Gustavson F G and Hachtel G D. A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formulas, Proc. IEEE, 1972, 60(1):

98～108

[11] 熊光楞等. 控制系统仿真与模型处理. 北京：科学出版社，1993

[12] Gear C W, Osterby O. Solving Ordinary Differential Equations with Discontinuities. ACMTOMS, 1984

[13] Shampine L F, et al. Detecting Stiffness with the Fehlberg (4, 5) Formulas. Computers and Mathematics with application. Vol. 3 No. 1, 1977

[14] 李伯虎等. 含不连续特性的非线性系统的数字仿真方法. 信息与控制，1983

[15] Ellison D. Efficient Automatic Integration of Ordinary Differential Equations with Discontinuities. Math, Computer. Simulation ⅩⅩⅢ(1981)

[16] Carver M B. Efficient Integration over Discontinuities in Ordinary Differential Equations Simulation, Math, Computer. Simulation ⅩⅩ(1978)

[17] Howe R W. New Method for handling Discontinuous Nonlinear Functions in Digital of Dynamic System. Proceedings of Summer Computer Simulation Conference, 1978

第 4 章
离散相似法

用数字计算机对一个连续系统进行仿真时，必须将这个系统看作一个时间离散系统。也就是说，我们只能计算各状态量在各计算步距点上的数值，它们是一些时间离散点的数值。第 3 章主要是从数值积分法的角度来讨论数字仿真问题，史密斯从控制和工程的概念出发提出离散相似问题，并导出离散相似法。

“离散相似法”就是将一个连续系统进行离散化处理，然后求得与它等价的离散模型。由于连续系统的模型可以用传递函数来表示，也可以用状态空间模型来表示，因此，与连续系统等价的离散模型可以通过两个途径获得，其一是对传递函数做离散化处理得到离散传递函数（或脉冲传递函数），称为频域离散相似模型；其二是基于状态方程离散化，得到时域离散相似模型。本章首先介绍时域离散相似法，然后介绍频域离散相似法，最后给出一个例子以说明其应用。

4.1 时域离散相似法

4.1.1 基本原理

假设有一个连续系统，它由以下状态方程描述

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{4.1}$$

对式(4.1)描述的连续系统进行离散化处理，如图 4.1 所示。在系统的输入端加上虚拟采样开关和虚拟信号重构器，输出端加一个虚拟采样开关。虚拟采样周期为 T 且同步。其中，$u(t)$是系统输入；$u(k)$是加虚拟采样开关后，在 kT 时刻的系统输入；$\boldsymbol{x}(k)$是加虚拟采样开关后在 kT 时刻系统输出；$\bar{u}(t)$、$\bar{\boldsymbol{x}}(t)$是等价的连续信号。只要$\bar{u}(t)$能足够精确地表示 $u(t)$，那么$\bar{\boldsymbol{x}}(t)$也就能足够精确地表示 $\boldsymbol{x}(t)$，这样，就能获得与连续系统等价的时域离散相似模型。

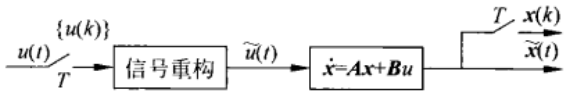


图 4.1 连续系统的离散化处理

对该连续系统进行离散化处理后可以得到系统离散相似模型如式(4.2)所示。

$$\boldsymbol{x}[(k+1)T] = \boldsymbol{\Phi}(T)\boldsymbol{x}(kT) + \boldsymbol{\Phi}_{\mathrm{m}}(T)u(kT) + \hat{\boldsymbol{\Phi}}_{\mathrm{m}}(T)\,\dot{u}(kT) \tag{4.2}$$

其中 T 是采样时间间隔(或称采样周期)；$u(k)$、$\boldsymbol{x}(k)$为系统 kT 时刻的输入及状态量；$\boldsymbol{\Phi}(T)$、$\boldsymbol{\Phi}_{\mathrm{m}}(T)$、$\hat{\boldsymbol{\Phi}}_{\mathrm{m}}(T)$为离散化后与系统模型有关的系数。下面将讨论怎样从式(4.1)经离散化处理获得式(4.2)所示的时域离散相似模型。

将方程两边取拉氏变换，经整理得到

$$\boldsymbol{X}(s) = (s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{x}(0) + (s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}U(s) \tag{4.3}$$

设

$$\boldsymbol{\Phi}(t) = \mathcal{L}^{-1}[(s\boldsymbol{I} - \boldsymbol{A})^{-1}] \tag{4.4}$$

则$\boldsymbol{\Phi}(t)=\mathrm{e}^{\boldsymbol{A}t}$称作状态转移矩阵。将式(4.4)代入式(4.3)，可得到

$$\boldsymbol{X}(s) = \mathcal{L}[\boldsymbol{\Phi}(t)]\boldsymbol{x}(0) + \mathcal{L}[\boldsymbol{\Phi}(t)]\boldsymbol{B}U(s) \tag{4.5}$$

由卷积公式 $\mathcal{L}\left[\int_0^t f(\tau)g(t-\tau)\mathrm{d}\tau\right] = F(s)G(s)$，称 $F(s)G(s)$为 f 和 g 的卷积，可以写作 $f * g$，其中，$\mathcal{L}[f(t)]=F(s)$，$\mathcal{L}[g(t)]=G(s)$。对式(4.5)取拉氏反变换，运用卷积公式得

$$\boldsymbol{x}(t) = \mathrm{e}^{\boldsymbol{A}t}\boldsymbol{x}(0) + \int_0^t \mathrm{e}^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)\mathrm{d}\tau \tag{4.6}$$

对离散化处理后的系统，设 kT 及$(k+1)T$ 为两个依次相连的采样时刻，则有

$$\boldsymbol{x}(kT) = \mathrm{e}^{\boldsymbol{A}kT}\boldsymbol{x}(0) + \int_0^{kT} \mathrm{e}^{\boldsymbol{A}(kT-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)\mathrm{d}\tau \tag{4.7}$$

$$\boldsymbol{x}[(k+1)T] = \mathrm{e}^{\boldsymbol{A}(k+1)T}\boldsymbol{x}(0) + \int_0^{(k+1)T} \mathrm{e}^{\boldsymbol{A}[(k+1)T-\tau]}\boldsymbol{B}\boldsymbol{u}(\tau)\mathrm{d}\tau \tag{4.8}$$

将式(4.8)一式(4.7)$\times \mathrm{e}^{\boldsymbol{A}T}$，可得

$$\boldsymbol{x}[(k+1)T] = \mathrm{e}^{\boldsymbol{A}T}\boldsymbol{x}(kT) + \int_{kT}^{(k+1)T} \mathrm{e}^{\boldsymbol{A}[(k+1)T-\tau]}\boldsymbol{B}\boldsymbol{u}(\tau)\mathrm{d}\tau \tag{4.9}$$

由于式(4.9)右端的积分与 k 无关，故可令 $k=0$。若信号重构器使 kT 与$(k+1)T$ 之间的 $u(t)$不变，即积分式中的 $u(\tau)$保持常数 $u(\tau)=u(kT)$，那么，式(4.9)可改写为

$$\begin{aligned}\boldsymbol{x}[(k+1)T] &= \mathrm{e}^{\boldsymbol{A}T}\boldsymbol{x}(kT) + \int_0^T \mathrm{e}^{\boldsymbol{A}(T-\tau)}\boldsymbol{B}\mathrm{d}\tau u(kT) \\ &= \boldsymbol{\Phi}(T)\boldsymbol{x}(kT) + \int_0^T \boldsymbol{\Phi}(T-\tau)\boldsymbol{B}\mathrm{d}\tau u(kT)\end{aligned} \tag{4.10}$$

若令 $\int_0^T \boldsymbol{\Phi}(T-\tau)\boldsymbol{B}\mathrm{d}\tau = \boldsymbol{\Phi}_{\mathrm{m}}(T)$，则有

$$\boldsymbol{x}[(k+1)T] = \boldsymbol{\Phi}(T)\boldsymbol{x}(kT) + \boldsymbol{\Phi}_{\mathrm{m}}(T)u(kT) \tag{4.11}$$

这就是一个离散状态方程。

式(4.11)是在假定信号重构器使输入量 $u(t)$在两个采样时刻之间保持不变的前提下

推导出来的，如图 4.2 中的矩形近似。实际上，输入量在两个采样时刻之间是变化的，这样就会引起误差。为了减小误差，可以假定在两个采样时刻之间，信号重构器使 $u(\tau)$ 为一斜坡函数，即用图 4.2 中的梯形近似。此时，在 kT 与 $(k+1)T$ 之间，相对矩形近似有一个 $\Delta u_k(\tau)$ 存在

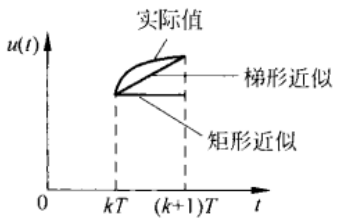


图 4.2 矩形近似与梯形近似

$$\Delta u_k(\tau) = \frac{u[(k+1)T] - u(kT)}{T}\tau \approx \dot{u}(kT)\tau \quad (4.12)$$

对应 $\Delta u_k(\tau)$，对 $\boldsymbol{x}[(k+1)T]$ 引起的变化量为

$$\Delta \boldsymbol{x}[(k+1)T] = \int_0^T \mathrm{e}^{\boldsymbol{A}(T-\tau)}\boldsymbol{B}\Delta u_k(\tau)\mathrm{d}\tau \approx \int_0^T \tau \mathrm{e}^{\boldsymbol{A}(T-\tau)}\boldsymbol{B}\mathrm{d}\tau\, \dot{u}(kT) \quad (4.13)$$

若令 $\int_0^\tau \tau \mathrm{e}^{\boldsymbol{A}(T-\tau)}\boldsymbol{B}\mathrm{d}\tau = \hat{\boldsymbol{\Phi}}_{\mathrm{m}}(T)$，则整个离散状态方程为

$$\boldsymbol{x}[(k+1)T] = \boldsymbol{\Phi}(T)\boldsymbol{x}(kT) + \boldsymbol{\Phi}_{\mathrm{m}}(T)u(kT) + \hat{\boldsymbol{\Phi}}_{\mathrm{m}}(T)\dot{u}(kT) \quad (4.14)$$

至此，得到式(4.2)中与系统模型有关的系数为

状态转移矩阵

$$\boldsymbol{\Phi}(T) = \mathrm{e}^{\boldsymbol{A}T}$$

输入信号采用零阶重构器引入的系数矩阵

$$\boldsymbol{\Phi}_{\mathrm{m}}(T) = \int_0^T \boldsymbol{\Phi}(T-\tau)\boldsymbol{B}\mathrm{d}\tau$$

输入信号采用三角形信号重构器后叠加的系数矩阵

$$\hat{\boldsymbol{\Phi}}_{\mathrm{m}}(T) = \int_0^T \tau\, \mathrm{e}^{\boldsymbol{A}(T-\tau)}\boldsymbol{B}\mathrm{d}\tau$$

下面举一个例子来说明如何用离散相似法来进行数字仿真。

例 4.1 假设有一系统，如图 4.3 所示。由图可知，该系统的开环部分是由一个非线性环节(饱和环节)及一个线性环节 $\dfrac{K}{s(s+1)}$ 所组成。今将线性部分转换成时域离散相似模型。

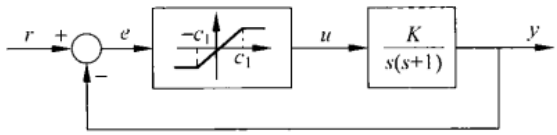


图 4.3 被仿真的系统结构图

已知线性部分状态方程为

$$\begin{cases} \boldsymbol{x} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u \\ y = \boldsymbol{C}\boldsymbol{x} \end{cases} \quad (4.15)$$

其中

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} K \\ 0 \end{bmatrix}, \quad \boldsymbol{C} = [0 \quad 1]$$

根据前述

$$\boldsymbol{\Phi}(T) = \mathrm{e}^{\boldsymbol{A}T} = \mathcal{L}^{-1}[(s\boldsymbol{I} - \boldsymbol{A})^{-1}]$$

因为 $s\boldsymbol{I}-\boldsymbol{A}=\begin{bmatrix} s & 0 \\ -1 & s+1 \end{bmatrix}$，所以有

$$(s\boldsymbol{I}-\boldsymbol{A})^{-1}=\begin{bmatrix} \dfrac{1}{s} & 0 \\ \dfrac{1}{s(s+1)} & \dfrac{1}{s+1} \end{bmatrix}$$

$$\boldsymbol{\Phi}(T)=\mathcal{L}^{-1}[(s\boldsymbol{I}-\boldsymbol{A})^{-1}]=\begin{bmatrix} 1 & 0 \\ 1-\mathrm{e}^{-T} & \mathrm{e}^{-T} \end{bmatrix}$$

$$\boldsymbol{\Phi}_{\mathrm{m}}(T)=\int_0^T \boldsymbol{\Phi}(T-\tau)\boldsymbol{B}\mathrm{d}\tau=\int_0^T \begin{bmatrix} 1 & 0 \\ 1-\mathrm{e}^{-(T-\tau)} & \mathrm{e}^{-\tau} \end{bmatrix}\begin{bmatrix} K \\ 0 \end{bmatrix}\mathrm{d}\tau$$

$$=\int_0^T \begin{bmatrix} K \\ K(1-\mathrm{e}^{-(T-\tau)}) \end{bmatrix}\mathrm{d}\tau=\begin{bmatrix} KT \\ K(T+1-\mathrm{e}^{-T}) \end{bmatrix}$$

则由式(4.14)可得线性动态环节的离散状态方程

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix}=\begin{bmatrix} 1 & 0 \\ 1-\mathrm{e}^{-T} & \mathrm{e}^{-T} \end{bmatrix}\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}+\begin{bmatrix} KT \\ K(T+1-\mathrm{e}^{-T}) \end{bmatrix}u(k) \tag{4.16}$$

式中$(k+1)$表示$[(k+1)T]$，(k)表示(kT)，以后为简单起见均如此表示。

利用式(4.16)即可进行数字仿真，仿真程序的框图如图 4.4 所示。用此程序可研究放大倍数 K 对系统输出响应的影响。例如，设输入信号 $r(t)$为阶跃函数，幅度为 25，且 $c_1=20$，并选采样周期 $T=0.05\mathrm{s}$，则可得 $K=1$ 及 $K=2$ 时的输出响应，如表 4.1 所示。

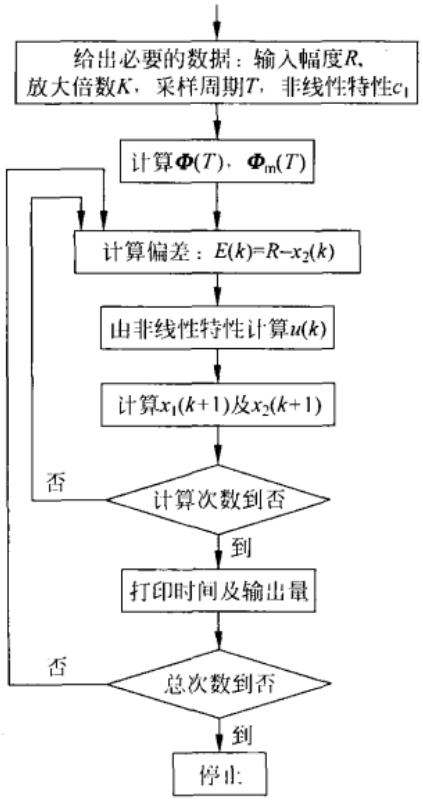


图 4.4 用离散相似法进行数字仿真

表 4.1 仿真结果

t	$y(K=1)$	$y(K=2)$
0	0	0
1	7.956 49	15.173 9
2	20.382 4	30.599 9
3	27.477 7	30.113
4	28.609 1	24.578 3
5	26.927 8	23.087 4
6	25.214 2	24.513 3
7	24.476 4	25.54
8	24.513 6	25.311 1
9	24.807 2	24.904 1
10	25.023 5	24.862 7
11	25.087 5	24.992 1
12	25.06	25.045 6
13	25.015 5	25.016 4
14	24.990 6	24.989
15	24.987 1	24.990 9

■

与第 3 章介绍的数值积分方法比较，方程系数$\boldsymbol{\Phi}(T)$，$\boldsymbol{\Phi}_{\mathrm{m}}(T)$，$\hat{\boldsymbol{\Phi}}_{\mathrm{m}}(T)$可以一次求出，这样每做一步积分只要计算一次右端函数。而 4 阶龙格-库塔数值积分方法，每做一步积分要计算 4 次右端函数。显然，用离散相似方法做仿真运算比用数值积分法做仿真运算运算量小，对于同一系统做仿真时，离散相似方法运算速度快。

由本小节所介绍的离散相似法原理可知，离散相似法的基本问题是计算$\boldsymbol{\Phi}(T)$，$\boldsymbol{\Phi}_{\mathrm{m}}(T)$，$\hat{\boldsymbol{\Phi}}_{\mathrm{m}}(T)$，而它们的计算主要归结为关于 $\mathrm{e}^{\boldsymbol{A}T}$的计算。下面我们将进一步讨论这一问题。

4.1.2 状态转移矩阵的计算

离散相似方法中的一个问题是状态转移矩阵 $\mathrm{e}^{\boldsymbol{A}T}$的计算。一些常用的 1 阶、2 阶环节，它们的$\boldsymbol{\Phi}(T)$，$\boldsymbol{\Phi}_{\mathrm{m}}(T)$，$\hat{\boldsymbol{\Phi}}_{\mathrm{m}}(T)$可以很方便地通过直接解析的方法来求解。对于高阶系统，若要求得到系统的离散状态方程，$\mathrm{e}^{\boldsymbol{A}T}$必须利用数值计算才能得到。

1. 泰勒级数展开法

$\mathrm{e}^{\boldsymbol{A}T}$的算法很多，但是用得最多的还是泰勒级数展开方法，即将 $\mathrm{e}^{\boldsymbol{A}T}$展成泰勒级数形式，然后截取其中的前若干项。这个方法首先是由 Lion 提出的。状态转移矩阵 $\mathrm{e}^{\boldsymbol{A}T}$可以用无穷级数来表示，如下所示

$$\mathrm{e}^{\boldsymbol{A}T}=\sum_{i=0}^{\infty}\frac{\boldsymbol{A}^i T^i}{i!},\quad \boldsymbol{A}^0=\boldsymbol{I} \tag{4.17}$$

这个级数在有限的间隔内是均匀收敛的，因此可以在一定精度内计算 $\mathrm{e}^{\boldsymbol{A}T}$。若级数在 $i=L$

处截断，则可以写成

$$e^{\boldsymbol{A}T}=\sum_{i=0}^{L}\frac{\boldsymbol{A}^iT^i}{i!}+\sum_{i=L+1}^{\infty}\frac{\boldsymbol{A}^iT^i}{i!}=\boldsymbol{M}+\boldsymbol{R} \tag{4.18}$$

式(4.18)中的第一项为级数的近似值，而第二项为余数项。要求

$$|r_{ij}|\leqslant E|m_{ij}| \quad E=10^{-d},d\text{ 为正整数}$$

或

$$|r_{\max}|\leqslant E_{\min} \tag{4.19}$$

其中，r_{ij} 和 m_{ij} 对应为 $\boldsymbol{R}$ 与 $\boldsymbol{M}$ 的元素，$r_{\max}$ 为 r_{ij} 中最大元素。式中 $E_{\min}$ 是容易求出的，但 $r_{\max}$ 却无法求出，因为 $\boldsymbol{R}$ 仍是一个无穷项的和。下面来估计 r_{max}。

令 $\|\boldsymbol{R}\|$ 为矩阵 $\boldsymbol{R}$ 的范数，根据矩阵范数的定义，$\|\boldsymbol{R}\|=\sum_{i,j=1}^{n}|r_{ij}|$，有 $r_{\max}\leqslant\|\boldsymbol{R}\|$。由

$$\begin{aligned}\|\boldsymbol{R}\|&=\left\|\sum_{i=L+1}^{\infty}\frac{\boldsymbol{A}^iT^i}{i!}\right\|\leqslant\sum_{i=L+1}^{\infty}\frac{\|\boldsymbol{A}\|^iT^i}{i!}\\&=\frac{\|\boldsymbol{A}\|^{L+1}\cdot T^{L+1}}{(L+1)!}\left(1+\frac{\|\boldsymbol{A}\|T}{L+2}+\frac{\|\boldsymbol{A}\|^2T^2}{(L+3)(L+2)}+\cdots\right)\\&\leqslant\frac{\|\boldsymbol{A}\|^{L+1}\cdot T^{L+1}}{(L+1)!}\left(1+\frac{\|\boldsymbol{A}\|T}{L+2}+\frac{\|\boldsymbol{A}\|^2T^2}{(L+2)^2}+\cdots\right)\end{aligned}$$

令

$$\frac{\|\boldsymbol{A}\|T}{L+2}=\varepsilon \tag{4.20}$$

则有

$$\|\boldsymbol{R}\|\leqslant\frac{\|\boldsymbol{A}\|^{L+1}\cdot T^{L+1}}{(L+1)!}(1+\varepsilon+\varepsilon^2+\cdots)$$

如果 $\varepsilon<1$，则

$$\|\boldsymbol{R}\|\leqslant\frac{\|\boldsymbol{A}\|^{L+1}\cdot T^{L+1}}{(L+1)!}\left(\frac{1}{1-\varepsilon}\right)$$

因此，若

$$\left|\frac{\|\boldsymbol{A}^{L+1}\cdot T^{L+1}\|}{(L+1)!}\cdot\frac{1}{1-\varepsilon}\right|\leqslant E_{\min} \tag{4.21}$$

则满足式(4.19)，$e^{\boldsymbol{A}T}$ 可以按照以下迭代过程来计算：

(1) 选择初始 L；

(2) 计算矩阵 $\boldsymbol{M}$ 及 $E_{\min}$；

(3) 用式(4.20)求 ε；

(4) 用式(4.21)判别是否满足精度的要求。若满足，则可用 $\boldsymbol{M}$ 来代替 $e^{\boldsymbol{A}T}$，否则 $L=L+1$，并重新计算(转(2))。

值得指出的是，在求出 $e^{\boldsymbol{A}T}$ 的同时，可以计算出式(4.2)中的另一项系数 $\boldsymbol{\Phi}_{\mathrm{m}}(T)$，因为 $\boldsymbol{\Phi}_{\mathrm{m}}(T)=\int_0^T e^{\boldsymbol{A}(T-\tau)}\boldsymbol{B}\mathrm{d}\tau$，令 $T-\tau=\tau'$，则有

$$\begin{aligned}\boldsymbol{\Phi}_{\mathrm{m}}(T)&=\int_0^T e^{\boldsymbol{A}\tau'}\boldsymbol{B}\mathrm{d}\tau'=\sum_{i=0}^{\infty}\int_0^T\frac{\boldsymbol{A}^i\tau'^i}{i!}\mathrm{d}\tau\boldsymbol{B}=\sum_{i=0}^{\infty}\frac{\boldsymbol{A}^iT^{i+1}}{(i+1)!}\boldsymbol{B}\\&=T\left(\sum_{i=0}^{\infty}\frac{\boldsymbol{A}^iT^i}{(i+1)!}\right)\boldsymbol{B}\end{aligned} \tag{4.22}$$

比较式(4.17)与式(4.22),可见$\boldsymbol{\Phi}_m(T)$的计算公式与e^{AT}的计算公式十分相似,因此可以编在一个计算程序中。同样,$\hat{\boldsymbol{\Phi}}_m(T)$的计算公式也可推导出来,读者不妨自行推导验证一下。

2. e^{AT}加速收敛算法

e^{AT}的计算要做大量的矩阵乘法运算,在有些情况下,泰勒级数展开法收敛性较差,即需要取很多项才能达到精度要求。然而项数增加,矩阵计算引入的舍入误差大大增加,影响计算精度。

以1阶系统为例,设系统微分方程为$\dot{y}=-ay(a>0)$,则$AT=-aT$,我们分别令aT等于0.1,0.5,1.0,2.0,泰勒展开式取前m项所达到的计算精度用10^{-b}表示,其结果如表4.2所示。

表4.2 不同aT下达到某一精度的项数

aT \ m / b	1	2	3	4	5	6	7	8	9	10
0.1	2	3	5	7	8	10	12	14	16	18
0.5	1	1	2	3	4	5	7	8	9	11
1.0		1	1	2	3	4	5	6	7	8
2.0				1	1	2	2	3	4	5

由表4.2中的数据可以看到,$aT<1$才有较好的收敛性。然而,在某些情况下,全部满足该条件比较困难(比如病态系统),因此,如何加速收敛就成为状态转移矩阵计算中一个必须解决的关键问题。下面介绍两种加速收敛的方法。

1) 等效转移法

若$|a_{ij}T|\gg 1$,考虑步长$T<1$,取时间比例尺$\alpha=1/T$,即原时间t,经等效转移后的时间为τ,则$\tau/t=1/T$。设原系统的拉氏变换为$F(s)$,等效转移后的系统拉氏变换为$F^*(s)$,由相似定理

$$F^*(s)=\alpha F(\alpha s)=\frac{1}{T}F\left(\frac{1}{T}s\right)$$

若

$$F(s)=\frac{b_1s^{n-1}+b_2s^{n-2}+\cdots+b_{n-1}s+b_n}{a_0s^n+a_1s^{n-1}+a_2s^{n-2}+\cdots+a_{n-1}s+a_n} \tag{4.23}$$

则

$$F^*(s)=\frac{1}{T}\frac{b_1\left(\frac{1}{T}\right)^{n-1}s^{n-1}+b_2\left(\frac{1}{T}\right)^{n-2}s^{n-2}+\cdots+b_{n-1}\left(\frac{1}{T}\right)s+b_n}{a_0\left(\frac{1}{T}\right)^{n}s^n+a_1\left(\frac{1}{T}\right)^{n-1}s^{n-1}+a_2\left(\frac{1}{T}\right)^{n-2}s^{n-2}+\cdots+a_{n-1}\left(\frac{1}{T}\right)s+a_n} \tag{4.24}$$

令

$$a_i^*=a_iT^{i+1},b_i^*=b_iT^i(i=0,1,2,\cdots,n) \tag{4.25}$$

并略加整理,可得

$$F^*(s)=\frac{b_1^*s^{n-1}+b_2^*s^{n-2}+\cdots+b_{n-1}^*s+b_n^*}{a_0^*s^n+a_1^*s^{n-1}+a_2^*s^{n-2}+\cdots+a_{n-1}^*s+a_n^*} \tag{4.26}$$

得到新的状态方程

$$\begin{cases}\dot{\boldsymbol{x}}^* = \boldsymbol{A}^* \boldsymbol{x}^* + \boldsymbol{B}^* \boldsymbol{u}^* \\ \boldsymbol{Y}^* = \boldsymbol{C}^* \boldsymbol{x}^*\end{cases} \tag{4.27}$$

显然，式(4.27) 中 $\boldsymbol{A}^*$ 阵的各元素 a_{ij}^* 与原来的 a_{ij} 相比较，缩小到 a_{ij} 的 $1/T$。设原来 $T=0.1$，转移后的系统 a_{ij}^* 为 a_{ij} 的 1/10。例如，仍设 $T=0.1$ 时，$a_{\max}T=5$，原状态转移矩阵的计算收敛速度显然十分缓慢，等效转移后 $a_{\max}^*T<0.5$，其状态转移矩阵计算的收敛性则大大加快。需要注意的是，仿真结果的时间比例尺也放大了 $1/T$ 倍。

2) 缩方与乘方

另一种加快收敛的方法是缩方与乘方法，下面介绍这种方法的原理。

根据 e^{AT} 的特性，若设 $T'=T\times 2^{-m}$，m 为大于零的整数，则有

$$e^{AT} = (e^{AT'\cdot 2^m}) = [(e^{AT'})^2]^m \tag{4.28}$$

由于 m 大于零，故 $T'<T$，假定 m 取得比较大，那么将有 $T'\ll T$。比如 $m=4$，则 $T'=\dfrac{T}{16}$，若先利用泰勒级数法来计算 $e^{AT'}$，那么可以取较少的级数项而能获得较高的精度，然后再利用式(4.28)将它进行 2^m 次方相乘，即可计算出 e^{AT}。

下面介绍有关这个算法的两个问题：

(1) 初始 L 的确定

由式(4.20)可知，当 $\dfrac{\|\boldsymbol{A}\|T'}{L+2}=\varepsilon<1$ 时收敛，考虑 $\|\boldsymbol{A}\|\leqslant n^2|a_{\max}|$，则可得

$$n^2 \mid a_{\max} \mid T' \leqslant L+2 \tag{4.29}$$

$$L \geqslant n^2 \mid a_{\max} \mid T' - 2 \tag{4.30}$$

经验表明，当 $a_{\max}T<1$ 时，一般初始 L 可取值为 n。

(2) 计算步骤

① 给定步距 T'，T 及误差限 E，比如，$E=10^{-6}$。

② 令 $L=n$，计算 $H=\sum\limits_{j=0}^{L}\dfrac{1}{j!}(\boldsymbol{A}T')^j$，并找出它的最小元素绝对值 $h_{\min}$，然后按式(4.21)估计 $\boldsymbol{R}_{\max}$。

③ 判断是否满足下式

$$\boldsymbol{R}_{\max} \leqslant E \cdot h_{\min} \tag{4.31}$$

若满足，则表示精度满足要求，转④，若不满足，则要加大 L，并重复②、③两步。

④ 计算 e^{AT}

$$e^{AT} = e^{ADT\cdot 2^m} \approx (\overbrace{(((((e^{AT})^2)^2)^2)\cdots)^2}^{m}) \tag{4.32}$$

需要指出的是，m 也不能太大，一般 m 应取值 4～8，否则计算 e^{AT} 时会产生很大的舍入误差。

4.1.3 增广矩阵法

4.1.1 节中介绍了用离散相似法对线性系统进行仿真的方法，对线性定常系统，有如下描述

$$\begin{cases}\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) \\ \boldsymbol{x}(t_0) = \boldsymbol{x}(0)\end{cases} \tag{4.33}$$

若加虚拟采样器及零阶信号重构器，则可得上述系统的等价离散模型，如式(4.34)所示。

$$\boldsymbol{x}(k+1) = \boldsymbol{\Phi}(T)\boldsymbol{x}(k) + \boldsymbol{\Phi}_{\mathrm{m}}(T)\boldsymbol{u}(k) \tag{4.34}$$

其中，T 为仿真步长，$\boldsymbol{\Phi}(T)=\mathrm{e}^{\boldsymbol{A}T}$，$\boldsymbol{\Phi}(T)=\int_0^T \mathrm{e}^{\boldsymbol{A}(T-\tau)}B\mathrm{d}\tau$。这种方法的误差来源于：

(1) $\mathrm{e}^{\boldsymbol{A}T}$ 的计算误差；

(2) 由于只考虑了 $\boldsymbol{\Phi}(T)$ 及 $\boldsymbol{\Phi}_{\mathrm{m}}(T)$，也即是近似认为 $\boldsymbol{u}(t)$ 在两个采样间隔中保持不变，所以也会引入误差。

尽管 $\boldsymbol{\Phi}(T)$ 及 $\boldsymbol{\Phi}_{\mathrm{m}}(T)$ 可归结为 $\mathrm{e}^{\boldsymbol{A}T}$ 的计算，而且 $\mathrm{e}^{\boldsymbol{A}T}$ 的计算误差可以通过缩方与乘方的方法减小，然而，虚拟采样后的信号器带来的误差却无法消除。为了提高仿真计算的精度，如果能将输入信号也作为系统的状态对待，那么只需要着眼于提高 $\mathrm{e}^{\boldsymbol{A}T}$ 的计算精度就能达到仿真精度的提高。基于这一思想，人们提出了增广矩阵法。

增广矩阵法是将式(4.33)所表示的系统输入量 $\boldsymbol{u}(t)$ 全部增广为系统的状态量，因而使 $\dot{\boldsymbol{x}}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{u}$ 这样一个非齐次常微分方程组转化为 $\dot{\tilde{\boldsymbol{x}}}=\tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}$ 这样一个齐次常微分方程组。这样式(4.33)改为式(4.35)所示的齐次方程组

$$\begin{cases}\dot{\tilde{\boldsymbol{x}}}(t) = \tilde{\boldsymbol{A}}\,\tilde{\boldsymbol{x}}(t) \\ \boldsymbol{y}(t) = \tilde{\boldsymbol{C}}\,\tilde{\boldsymbol{x}}(t) \\ \tilde{\boldsymbol{x}}(t_0) = \tilde{\boldsymbol{x}}(0)\end{cases} \tag{4.35}$$

其中，$\tilde{\boldsymbol{x}}(t)$ 既包含原有的状态量 $\boldsymbol{x}(t)$，还包含输入量 $\boldsymbol{u}(t)$。这样，与式(4.35)等价的离散模型就变成

$$\tilde{\boldsymbol{x}}(k+1) = \hat{\boldsymbol{\Phi}}(t)\,\tilde{\boldsymbol{x}}(k) \tag{4.36}$$

其中

$$\hat{\boldsymbol{\Phi}}(T) = \mathrm{e}^{\tilde{\boldsymbol{A}}T} \tag{4.37}$$

显然，利用式(4.36)来进行仿真将只有一项误差，就是计算 $\mathrm{e}^{\tilde{\boldsymbol{A}}T}$ 的误差。

$u(t)$ 为系统的输入量，假定是一个标量，也就是说，我们研究的系统是一个单输入系统。当 $u(t)$ 是一些典型函数时，上述增广的想法很容易实现。例如，阶跃输入时，设

$$u(t) = U_0 1(t)$$

则，定义第 $n+1$ 个状态变量为

$$x_{n+1}(t) = u(t) = U_0 1(t)$$

故

$$\dot{x}_{n+1}(t) = 0$$

根据式(4.33)及式(4.34)可得增广后的状态方程及输出方程为

$$\begin{bmatrix}\dot{\boldsymbol{x}}(t) \\ \dot{x}_{n+1}(t)\end{bmatrix} = \left[\begin{array}{c:c}\boldsymbol{A} & \boldsymbol{B} \\ \hdashline 0 & 0\end{array}\right]\begin{bmatrix}\boldsymbol{x}(t) \\ x_{n+1}(t)\end{bmatrix}$$

$$y(t) = [\boldsymbol{C} \vdots 0]\begin{bmatrix}\boldsymbol{x}(t) \\ x_{n+1}(t)\end{bmatrix}$$

初始条件

$$\begin{bmatrix} \boldsymbol{x}(0) \\ \cdots\cdots \\ x_{n+1}(0) \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_0 \\ \cdots \\ U_0 \end{bmatrix}$$

考虑一般情形，设作用函数 $u(t)$ 可以表示成如下形式

$$U(s) = d\,\frac{p_1 + p_2 s + \cdots + p_l s^{l-1} + s^l}{q_1 + q_2 s + \cdots + q_m s^{m-1} + s^m} \tag{4.38}$$

其中 $m>l$，则可视 $u(t)$ 为上述 m 阶系统的自由响应。设该 m 阶系统的状态变量为 $\boldsymbol{x}_u$，不难得到如下方程

$$\dot{\boldsymbol{x}}_u = \boldsymbol{A}_u \boldsymbol{x}_u \tag{4.39}$$

$$y_u(t) = u(t) = \boldsymbol{C}_u \boldsymbol{x}_u \tag{4.40}$$

其中

$$\boldsymbol{x}_u(t_0) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad \boldsymbol{A}_u = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ & & & \ddots & & \vdots \\ & & & & \cdots & 1 \\ -q_1 & -q_2 & -q_3 & \cdots & -q_{m-1} & -q_m \end{bmatrix} \tag{4.41}$$

$$\boldsymbol{C}_u = (\underbrace{dp_1, dp_2, \cdots, dp_l}_{l}, \underbrace{d, 0, \cdots, 0}_{m-l}) \tag{4.42}$$

将该 m 阶系统增广到原 n 阶系统，组成的增广状态方程如下

$$\tilde{\boldsymbol{x}} = \underbrace{\begin{bmatrix} \boldsymbol{x} \\ \cdots \\ \boldsymbol{x}_u \end{bmatrix}}_{(m+n)\times 1} \quad \tilde{\boldsymbol{A}} = \underbrace{\begin{bmatrix} \overbrace{\boldsymbol{A}}^{n\times n} & \overbrace{\boldsymbol{B}\boldsymbol{C}_u}^{n\times m} \\ \underbrace{0}_{m\times n} & \underbrace{\boldsymbol{A}_u}_{m\times m} \end{bmatrix}}_{(m+n)\times(m+n)} \quad \tilde{\boldsymbol{x}}(0) = \begin{bmatrix} \boldsymbol{x}(0) \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{4.43}$$

不难画出增广系统的结构图如图 4.5 所示。

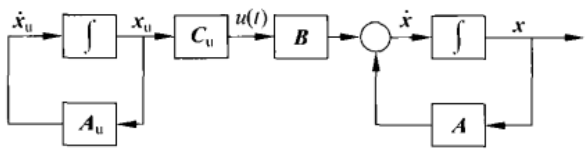


图 4.5 增广系统结构图

值得注意的是，系统阶次由原来的 n 阶扩充到 $n+m$ 阶，因而状态转移矩阵的计算工作量将加大。

4.2 面向结构图的非线性系统仿真

由 4.1.1 节可知，对于线性系统采用离散相似方法进行仿真可归结为计算式(4.11)所示的离散状态方程，其中主要是计算 $\boldsymbol{\Phi}(T) = \mathrm{e}^{\boldsymbol{A}T}$ 及 $\boldsymbol{\Phi}_m(T) = \int_0^T \mathrm{e}^{\boldsymbol{A}(T-\tau)}\boldsymbol{B}\mathrm{d}\tau$。由于线性系统的数学模型可以写成 1 阶线性微分方程组(即状态方程)的形式，所以可以用数值计算的方

法事先计算出整个系统的$\boldsymbol{\Phi}(T)$及$\boldsymbol{\Phi}_m(T)$，然后利用离散状态方程来进行仿真。

然而，实际系统中往往存在一些非线性，在结构描述中它们也以典型环节的形式出现。由于这类非线性环节往往位于线性环节之间，不可能统一计算整个系统的$\boldsymbol{\Phi}(T)$及$\boldsymbol{\Phi}_m(T)$，而只能计算出各个线性部分的$\boldsymbol{\Phi}_i(T)$及$\boldsymbol{\Phi}_{m_i}(T)(i=1,2,\cdots,N$，$N$表示该系统共有$N$个线性部分)。这样，利用它们也只能计算出各线性部分的输出。如果线性环节之间存在非线性环节，为了计算下一步各线性部分的输入，或者将各线性环节的输入先进行位于其前面的非线性环节的计算，也可先将线性环节的输出进行位于其后的各非线性部分的计算。

基于这一基本思想，对用结构图描述的非线性系统，本节介绍一种用离散相似法实现系统仿真的方法。首先，在定义仿真模型时，选择如图4.6所示的环节作为基本环节。不难看出，该环节是在典型线性环节的前面与后面均附属了一个非线性环节，这样可以以被仿真的系统中的线性典型环节为基础确定系统的阶次和结构，包括环节个数、参数和连接矩阵等。还可以确定非线性环节的归属，例如，假设两线性环节之间仅有一个非线性环节，则该非线性环节既可视为归属于位于其前的线性环节，也可视为归属于位于其后的线性环节。对每类非线性环节给出其特征参数和类型参数，以及它位于某一环节前或后的描述参数。

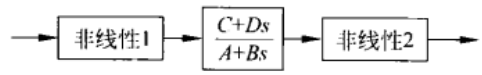


图4.6 一个完整的动态环节

计算过程分4步：

① 不考虑非线性，根据连接矩阵计算出各环节的输入；

② 根据各环节前的非线性计算出各环节中线性动态部分的输入；

③ 根据线性部分的离散状态方程计算各环节线性部分的输出；

④ 根据各环节后的非线性计算出各环节中线性动态部分的输出。

然后重复上述步骤，直到仿真结束。

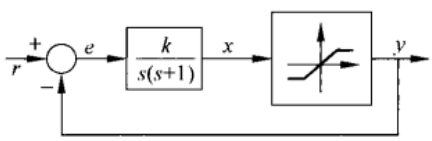


图4.7 非线性系统框图

例如，如图4.7所示的非线性系统。该系统有一个线性部分$k/[s(s+1)]$，有一个非线性部分，它用$y=f(x)$表示“饱和特性”。为了仿真该系统，首先可以将线性部分的$\boldsymbol{\Phi}(T)$及$\boldsymbol{\Phi}_m(T)$计算出来，然后按以下步骤计算不同时刻的输出$y(t)$。

(1) 选择采样周期T，给定$r(t)$及x的初值，计算y及e之值；

(2) 令$t=0$，$k=0$，按下式计算各点的值

$$x(k+1)=\boldsymbol{\Phi}(T)x(k)+\boldsymbol{\Phi}_m(T)e(k)$$

$$y(k+1)=f(x(k+1))$$

$$e(k+1)=r(k+1)-y(k+1)$$

(3) $k+1\to k$，重复步骤(2)，直到仿真结束。

本书附录中的CSS02.C就是这种思想的实现。该程序可用于多种系统的仿真。下面对该程序的实现技术作一说明。

4.2.1 典型线性动态环节$\boldsymbol{\Phi}(T)$，$\boldsymbol{\Phi}_m(T)$，$\hat{\boldsymbol{\Phi}}_m(T)$的计算

CSS02.C中包含的典型线性动态环节有积分、比例积分、惯性、超前(滞后)和比例5

种。均以$(C_i+D_is)/(A_i+B_is)$形式描述。根据每种情况的 A_i,B_i,C_i,D_i 取值，确定所对应的线性环节类型，进而计算出该环节的状态转移矩阵的值，具体如下：

(1) $B_i=0,D_i=0$ 为比例环节

$$K_i=\frac{C}{A}$$

(2) $A_i=0,D_i=0$ 为积分环节

$$\frac{C_i}{B_is}=\frac{K_i}{s}$$

(3) 仅 $A_i=0$ 为比例积分环节

$$\frac{C_i+D_is}{B_is}=\frac{K_i(b_is+1)}{s}$$

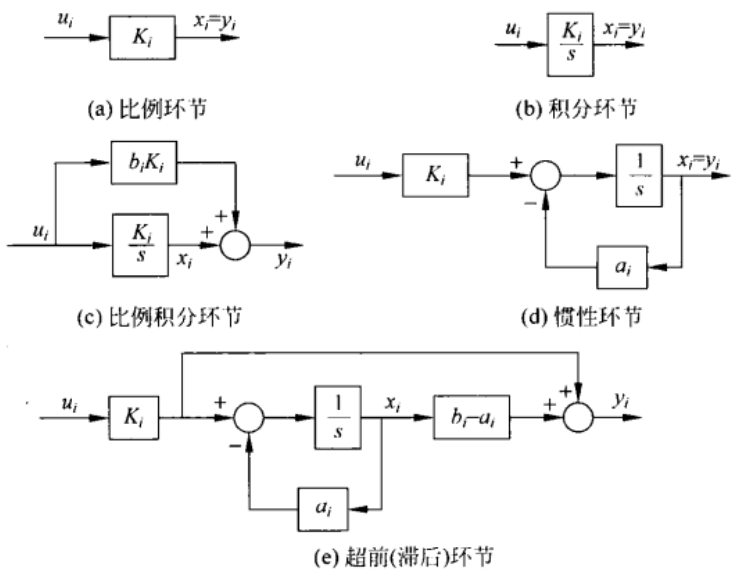
(4) 仅 $D_i=0$ 为惯性环节

$$\frac{C_i}{A_i+B_is}=\frac{K_i}{s+a_i}$$

(5) A_i,B_i,C_i,D_i 均不等于零，为超前(滞后)环节

$$K_i\frac{s+b_i}{s+a_i}$$

它们的结构图如图 4.8 所示。



(a) 比例环节　(b) 积分环节

(c) 比例积分环节　(d) 惯性环节

(e) 超前(滞后)环节

图 4.8　五种情况的结构图

可以分别列出相应的 5 套计算公式：

(1) 比例环节

$$y(k+1)=K_iu(k+1)$$

(2) 积分环节

$$x(k+1)=x(k)+K_iTu(k)+0.5K_iT^2\dot{u}(k)$$
$$y(k+1)=x(k+1)$$

(3) 比例积分环节

$$x(k+1) = x(k) + K_i T u(k) + 0.5 K_i T^2 \dot{u}(k)$$
$$y(k+1) = x(k+1) + b_i K_i u(k+1)$$

(4) 惯性环节

$$x(k+1) = e^{-a_i T} x(k) + \frac{K_i}{a_i}(1 - e^{-a_i T}) u(k) + \left[\frac{K_i}{a_i^2}(e^{-a_i T} - 1) + \frac{K_i}{a_i} T\right] \dot{u}(k)$$
$$y(k+1) = x(k+1)$$

(5) 超前(滞后)环节

$$x(k+1) = e^{-a_i T} x(k) + \frac{K_i}{a_i}(1 - e^{-a_i T}) u(k) + \left[\frac{K_i}{a_i^2}(e^{-a_i T} - 1) + \frac{K_i}{a_i} T\right] \dot{u}(k)$$
$$y(k+1) = (b_i - a_i) x(k+1) + K_i u(k+1)$$

4.2.2 常用非线性环节计算子程序

常用非线性环节包括饱和非线性、失灵区非线性、回环非线性等环节。图 4.9～图 4.11 分别给出这些非线性环节计算子程序框图。图 4.11 中 u_i^0, u_o^0 为上一步输入、输出信号的数据。

(1) 饱和非线性环节

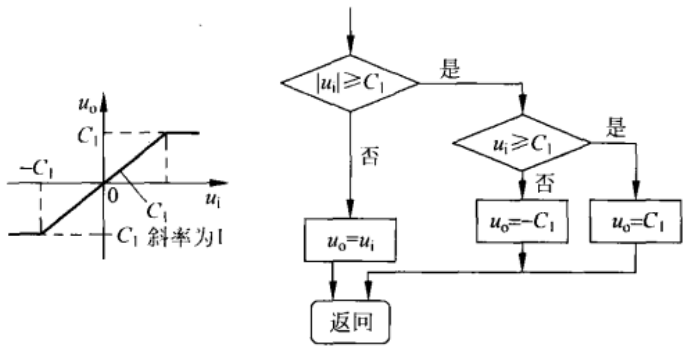


图 4.9 饱和非线性特性及子程序框图

(2) 失灵区非线性环节

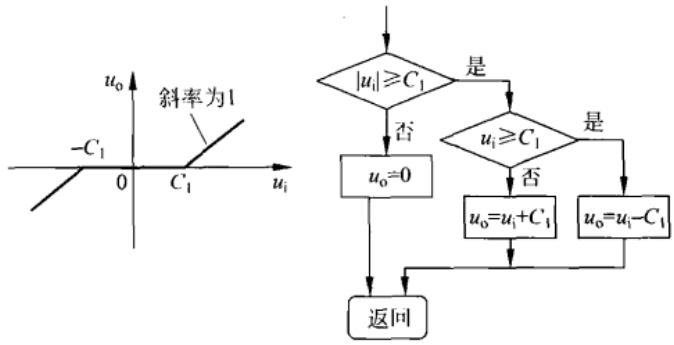


图 4.10 失灵区非线性特性及子程序框图

（3）回环非线性环节

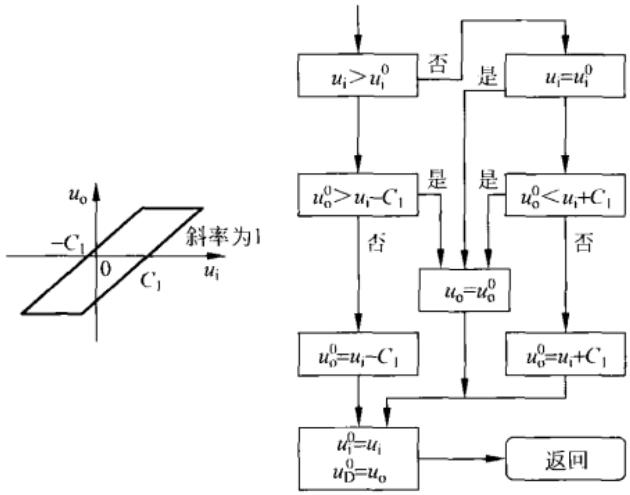


图 4.11 回环非线性特性及子程序框图

4.2.3 非线性系统离散相似法仿真程序 CSS02.C

CSS02.C 程序是面向结构图的，即系统结构及其参数以结构图的形式给出，采用离散相似法仿真。系统模型中可以包含饱和、失灵区及回环等非线性环节。CSS02.C 程序框图如图 4.12 所示。

该程序要求将系统中的线性动态环节写成$\frac{C_i + D_i s}{A_i + B_i s}$的形式，每个环节要输入 8 个参数：$A$，$B$,$C$,$D$,$X$,$Y$,$Z$,$S$,其中，$A$,$B$,$C$,$D$ 为环节的参数；X,Y 是这个环节的状态变量及输出变量的初值(X,Y 的定义如图 4.8 所示)；Z 及 S 表示附属非线性环节的类别及参数。程序规定：

$Z=0$ 表示该环节无非线性；

$Z=1$ 表示该环节前有饱和非线性；

$Z=2$ 表示该环节前有失灵区非线性；

$Z=3$ 表示该环节前有回环非线性；

$Z=4$ 表示该环节后有饱和非线性；

$Z=5$ 表示该环节后有失灵区非线性；

$Z=6$ 表示该环节后有回环非线性；

$S=C_1$ 参看图 4.9、图 4.10 和图 4.11。

程序中主要变量说明：

$STEP$ 计算步距；

L_1 打印间隔(即每隔 $L_1 \times STEP$ 打印一点)；

L_2 打印点数(即仿真总时间为 $STEP \times L_1 \times L_2$)；

N 系统的环节数；

R 输入阶跃函数的幅度；

N_0 打印变量的个数；

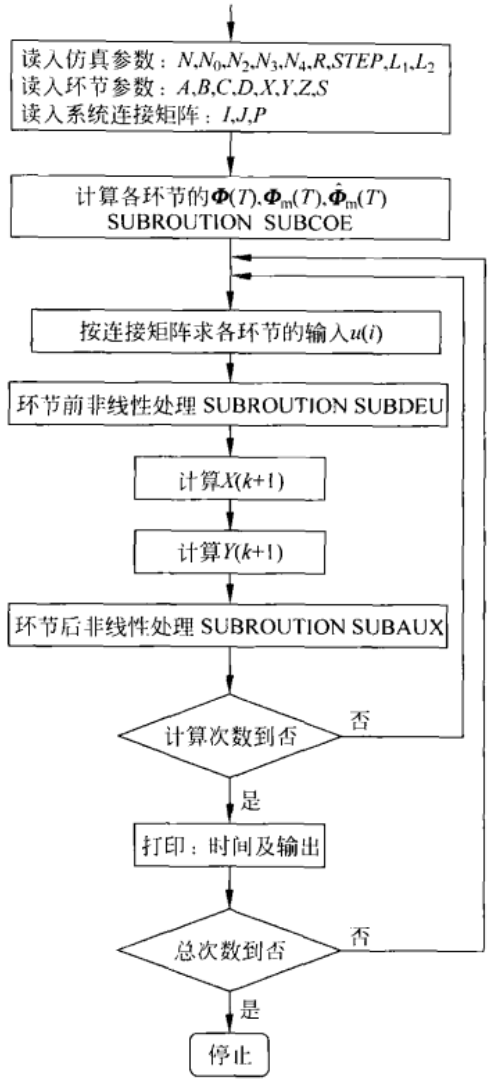


图 4.12 CSS02 程序框图

N_2
N_3 打印变量的编号。
N_4

下面以图 4.13 所示的小功率随动系统为例介绍 CSS02.C 的使用，仿真步骤如下：

(1) 将该系统中各线性动态环节编号，其中非线性环节看作第 4 号环节的附属环节，故系统共有 4 个环节。

(2) 输入 CSS02 仿真程序。

(3) 运行 CSS02，按屏幕上的提示回答问题：

The order of system $N=4$

Input $N0$, $N2$, $N3$, $N4=3$, 1, 3, 4

Enter the input $R=10$

Enter the $STEP$

0.025

Enter $L1$, $L2$=

10, 60

Enter $A(1)$, $B(1)$, $C(1)$, $D(1)$, $X(1)$, $Y(1)$, $Z(1)$, $S(1)$

0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0

Enter $A(2)$, $B(2)$, $C(2)$, $D(2)$, $X(2)$, $Y(2)$, $Z(2)$, $S(2)$

0.0, 1.0, 0.5, 1.0, 0.0, 0.0, 0.0, 0.0

Enter $A(3)$, $B(3)$, $C(3)$, $D(3)$, $X(3)$, $Y(3)$, $Z(3)$, $S(3)$

2.0, 1.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0

Enter $A(4)$, $B(4)$, $C(4)$, $D(4)$, $X(4)$, $Y(4)$, $Z(4)$, $S(4)$

10.0, 1.0, 10.0, 0.0, 0.0, 6.0, 1.0

Enter Link matrix I, J, P=

1, 0, 1.0, 1, 4, −1.0, 2, 1, 1.0, 3, 2, 1.0, 4, 3

1.0, 0, 0, 0.0

(4) 终端上将显示仿真结果，仿真结束时可由打印机将其一次输出。仿真结果如表 4.3 所示。仿真结果表明，由于系统中有回环非线性，系统会产生自振荡，当 $C_1=1$ 时，自振荡的振幅为 9.4～10.8，振荡周期约为 6s。

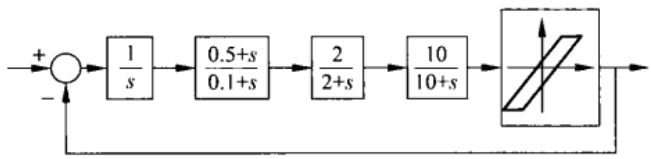


图 4.13 小功率随动系统结构图

表 4.3 小功率随动系统仿真结果

T	$Y(1)$	$Y(3)$	$Y(4)$
0	0	0	0
0.25	2.75	0.595 332	1
0.5	5.237 67	2.105 52	0.465 295
0.75	7.402 11	4.158 26	2.321 66
1	9.051 67	6.431 27	6.789 9
1.25	10.136 6	8.677 54	6.789 9
1.5	10.667 5	10.717 7	8.919 96
1.75	10.697 9	12.434 9	10.770 5
2	10.309 7	13.766 7	12.257 9
2.25	9.599	14.695 3	13.347 1
2.5	8.665 83	15.237 3	14.041 5
2.75	7.605 48	15.434	14.373 1
3	6.502 09	15.342 2	14.418 6
3.25	5.397 45	15.022 7	14.418 6
3.5	4.292 82	14.518 3	14.418 6
3.75	3.188 18	13.855 8	14.418 6
4	2.083 54	13.052 6	14.381 5

续表

T	$Y(1)$	$Y(3)$	$Y(4)$
4.25	1.091 99	12.132 5	13.503 5
4.5	0.333 347	11.185	12.561 8
4.75	−190 555	10.286 3	11.640 2
5	−492 995	9.489 32	10.800 3
5.25	−600 024	8.827 14	10.832
5.5	−545 528	8.315 32	9.511 01
5.75	−366 964	7.955 25	9.090 72
6.000 01	−101 898	7.737 74	8.816 97
6.250 01	0.214 622	7.449 4	8.675 59
6.500 01	0.551 89	7.656 02	8.644 5
6.750 01	0.890 765	7.747 59	8.644 5
7.000 01	1.229 64	7.902 16	8.644 5
7.250 01	1.568 52	8.108 96	8.644 5
7.500 01	1.907 39	8.361 11	8.644 5
7.750 01	2.246 27	8.654 1	8.644 5
8.000 01	2.585 14	8.984 87	8.644 5
8.250 01	2.924 02	9.351 23	8.644 5
8.500 2	3.262 9	0.751 54	8.644 5
8.750 02	3.564 47	10.187 1	9.009 12
9.000 02	3.758 27	10.605 8	9.438 32
9.250 02	3.847 49	10.989 2	9.839 38
9.500 02	3.842 73	11.31	10.186
9.750 02	3.759 77	11.557	10.462 8
10	3.617 06	11.727 2	10.663 6
10.25	3.433 63	11.824 2	10.789 4
10.5	3.227 53	11.856	10.846 6
10.75	3.014 5	11.833 6	10.852 4
11	2.801 41	11.768 5	10.852 4
11.25	2.588 31	11.668 5	10.852 4
11.5	2.375 21	11.538 4	10.852 4
11.75	2.162 11	11.381 5	10.852 4
12	1.949 01	11.199 9	10.852 4
12.75	1.735 91	10.995	10.852 4
12.5	1.522 81	10.768 1	10.852 4
12.75	1.309 71	10.52	10.852 4
13	1.096 61	10.253 1	10.852 4
13.25	0.883 511	9.962 81	10.852 4
13.5	0.671 282	9.655 01	10.779 2
13.75	0.515 799	9.336 84	10.463 1
14	0.438 969	9.039 29	10.156
14.45	0.435 229	8.782 89	9.882 37
14.5	0.494 087	8.579 65	9.657 52
14.75	0.602 213	8.434 41	9.489 08
15	0.745 199	8.346 32	9.378 39

4.3 频域离散相似法

我们知道，一个连续系统的模型，除了状态方程外，传递函数也是描述形式之一。如果直接面向 s 域的传递函数 $G(s)$，根据相似原理，得到与它相匹配的 z 域传递函数 $G(z)$，从而导出其差分模型。

这里的所谓“匹配”，既包括动态性能的匹配也包括稳态性能的匹配。也就是说，$G(s)$ 的零点、极点要与相应 $G(z)$ 的零、极点匹配，而且对于同一个输入函数，由 $G(s)$ 及 $G(z)$ 所求出的输出函数应具有相同的稳态特征，比如说终值相同等。

将离散相似法用于连续传递函数从而得到系统离散传递函数（或脉冲传递函数），这种方法称作频域离散相似方法。

4.3.1 替换法

从 $G(s)$ 直接导出与它相匹配的 $G(z)$ 有两种方法，一种是替换法，即设法找到 s 与 z 的一个对应公式，然后将 $G(s)$ 中的 s 转换成 z，由此求得 $G(z)$；另一种是 4.3.2 节将要介绍的根匹配法，即设法找到一个 $G(z)$，它具有与 $G(s)$ 相同的零极点。

已知 z 与 s 的关系为 $z=\mathrm{e}^{sT}$，这是一个超越函数，在进行仿真模型变换时，不能直接用它来替换。下面介绍 s 与 z 之间的两种替换方法，即欧拉替换和双线性替换（图斯汀替换）。

1. 欧拉替换

今假定要求出能满足 $\mathrm{d}x/\mathrm{d}t=u(t)$ 这个微分方程的 $x(t)$，根据欧拉积分公式，可得

$$x_{k+1}=x_k+Tu_k=x_k+T\dot{x}_k \tag{4.44}$$

所以

$$(z-1)x=T\dot{x}$$

即

$$\frac{x}{\dot{x}}=\frac{T}{z-1}$$

因为 $\frac{x}{\dot{x}}=\frac{1}{s}$，故有

$$s^{-1}=\frac{T}{z-1}$$

即

$$s=\frac{z-1}{T} \tag{4.45}$$

式(4.45)称为欧拉替换，公式虽然很简单，但是并不实用。因为若用式(4.45)的关系代入 $G(s)$，由此获得的 $G(z)$，在 T 较大的情况下，将会使 $G(z)$ 不稳定。为了说明这一点，让我们来看一下式(4.45)所表示的 s 平面与 z 平面之间的映射关系。

已知，若 $G(z)$ 的所有极点全部都在 z 平面的单位圆内，则它所表示的系统就是稳定的。若将 z 平面的单位圆按照式(4.45)映射到 s 平面上，设

$$s = \sigma + j\omega$$

根据式(4.45)，有 $z = Ts + 1$，则

$$|z|^2 = (1 + T\sigma)^2 + \omega^2 T^2 \tag{4.46}$$

对于 z 平面上的单位圆，有 $|z|^2 = 1$，故

$$(1 + T\sigma)^2 + \omega^2 T^2 = 1$$

即

$$T^2\sigma^2 + 2T\sigma + \omega^2 T^2 = 0 \tag{4.47}$$

也就是

$$\sigma^2 + \frac{2}{T}\sigma + \omega^2 = 0$$

$$\left(\sigma + \frac{1}{T}\right)^2 + \omega^2 = \left(\frac{1}{T}\right)^2 \tag{4.48}$$

式(4.48)说明，在 s 平面上所表示的图形正好是以 $(-1/T, 0)$ 为圆心，以 $1/T$ 为半径的一个圆，如图 4.14 所示。

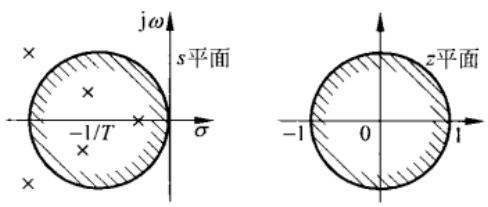


图 4.14 s 域到 z 域的映射关系

这就是说，z 平面上的单位圆按式(4.45)映射到 s 平面上，将是一个以 $(-1/T, 0)$ 为圆心，以 $1/T$ 为半径的圆。

假定 $G(s)$ 的极点如图 4.14 所示(图中的×)，显然，该系统是稳定的，其中有三个极点在 s 左半平面的单位圆内，而有两个虽在 s 左半平面，但位于单位圆外。若用式(4.45)来替换 $G(s)$ 中的 s，所得的 $G(z)$ 将有三个极点在单位圆内，两个在单位圆外，因此 $G(z)$ 是不稳定的。

一个原来稳定的系统 $G(s)$，通过式(4.45)的替换得到的仿真模型 $G(z)$ 却是不稳定的。显然，这种仿真模型是不能采用的。

由图 4.14 可知，为使 $G(z)$ 稳定，要求加大 $1/T$，也即减小 T。为保证在 $G(s)$ 稳定的条件下替换得到的 $G(z)$ 也能稳定，只有当 $T \to 0$(趋向 0)时才有可能，因此，这种替换具有很大的局限性。

2. 双线性替换

比较简单而且实用的公式是双线性替换公式，它可以从梯形积分公式中直接推导出来。按这种替换公式进行替换，可以保证 $G(z)$ 的稳定性，而且具有一定的仿真精度。

已知梯形积分公式为

$$x_{k+1} = x_k + \frac{T}{2}(\dot{x}_k + \dot{x}_{k+1})$$

即

$$(z - 1)x = \frac{T}{2}(z + 1)\dot{x}$$

则有

$$s^{-1}=\frac{T}{2}\frac{z+1}{z-1}$$

即

$$s=\frac{2}{T}\frac{z-1}{z+1} \tag{4.49}$$

式(4.49)称为双线性替换公式,也可写为

$$z=\frac{1+sT/2}{1-sT/2}$$

则

$$|z|^2=\frac{\left(1+\frac{\sigma T}{2}\right)^2+\left(\frac{\omega}{2}T\right)^2}{\left(1-\frac{\sigma T}{2}\right)^2+\left(\frac{\omega}{2}T\right)^2} \tag{4.50}$$

由式(4.50)可知,若 $\sigma<0$,则 $|z|<1$;若 $\sigma=0$,则 $|z|=1$;而若 $\sigma>0$,则 $|z|>1$。这就是说,若采用式(4.49)的替换公式,z 平面上的单位圆,映射到 s 平面上将是整个左半平面,其逆也真,如图 4.15 所示。

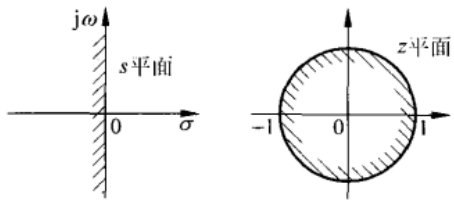


图 4.15 双线性替换的映射关系

因此,利用式(4.49)的替换公式,如果原来 $G(s)$稳定,那么 $G(z)$也是稳定的。例如,假定传递函数

$$G(s)=\frac{s}{(s+1)^2}=\frac{Y(s)}{U(s)} \tag{4.51}$$

则根据式(4.49)进行替换,得

$$G(z)=\frac{\frac{2}{T}\frac{z-1}{z+1}}{\left(\frac{2}{T}\frac{z-1}{z+1}+1\right)^2}=\frac{2T(z^2-1)}{[(2+T)z-(2-T)]^2}$$

$$=\frac{\frac{2T}{(2+T)^2}(z^2-1)}{\left[z-\left(\frac{2-T}{2+T}\right)\right]^2} \tag{4.52}$$

将式(4.52)写成差分方程式,得

$$y_k=2\left(\frac{2-T}{2+T}\right)y_{k-1}-\left(\frac{2-T}{2+T}\right)^2y_{k-2}+\frac{2T}{(2+T)^2}(u_k-u_{k-2}) \tag{4.53}$$

为了说明利用双线性替换公式(4.49)所得的仿真模型具有一定的精度,我们用频率分析法分析一下 $G(s)$与 $G(z)$两者之间的误差。

根据式(4.51)的 $G(s)$,用 $s=j\omega$ 代入,可得

$$G(j\omega)=\frac{j\omega}{(j\omega+1)^2}=\frac{2\omega^2+j(\omega-\omega^3)}{(1+\omega^2)^2} \tag{4.54}$$

它的幅频与相频的具体数据如表 4.4 所示。对于式(4.52)所示的 $G(z)$,同样可以用 $z=e^{j\omega T}$ 代入,得

$$G(j\omega)=\frac{2T(e^{2j\omega T}-1)}{[e^{j\omega T}(2+T)-(2-T)]^2} \tag{4.55}$$

用 $e^{j\omega T}=\cos\omega T+j\sin\omega T$ 代入，同样可以求出它的幅频特性与相频特性。当 $T=1s$ 时，它们的具体数值也列于表 4.4 中。

比较表 4.4 中的两组数据可知：当 $T=1s$ 时，由 $G(s)$ 与 $G(z)$ 所获得的频率特性在 $\omega=1rad/s$ 之内二者是十分接近的(幅频的误差仅 1.5%，相频的误差在 5°之内)。这说明，用双线性替换公式所获得的仿真模型不仅稳定，而且有一定的仿真精度。

表 4.4 用双线性替换法所获得的仿真模型与实际连续系统的比较

	ω/rad/s	0.1	0.3	0.6	0.8	1.0	1.2	1.5
幅频特性	连续系统	0.099 01	0.275 2	0.441 2	0.487 8	0.50	0.491 8	0.461 5
	双线性替换法	0.099 1	0.277	0.447	0.493	0.498	0.476	0.417
相频特性	连续系统	74.858°	56.60°	28.07°	12.68°	0°	−10.39°	−22.62°
	双线性替换法	78.57°	56.36°	26.51°	9.57°	−5.06°	−17.67°	−33.56°

双线性替换不仅可以获得精度较高的仿真模型，而且能利用计算机程序来实现这种替换。下面介绍一种程序替换算法。

设线性系统的传递函数为

$$G(s)=\frac{U(s)}{E(s)}=\frac{a_0s^n+a_1s^{n-1}+\cdots+a_{n-1}s+a_n}{b_0s^n+b_1s^{n-1}+\cdots+b_{n-1}s+b_n} \tag{4.56}$$

在双线性替换下得到的 z 传递函数为

$$G_0(z)=\frac{U(z)}{E(z)}=\frac{d_0z^n+d_1z^{n-1}+\cdots+d_{n-1}z+d_n}{e_0z^n+e_1z^{n-1}+\cdots+e_{n-1}z+e_n} \tag{4.57}$$

现需要由 $a_i,b_i(i=0,1,\cdots,n)$ 确定 $d_i,e_i(i=0,1,\cdots,n)$，若直接将双线性替换公式代入 $G_0(s)$，可得

$$G_0(z)=\frac{a_0\left(\frac{2}{T}\right)^n\left(\frac{z-1}{z+1}\right)^n+a_1\left(\frac{2}{T}\right)^{n-1}\left(\frac{z-1}{z+1}\right)^{n-1}+\cdots+a_{n-1}\frac{2}{T}\left(\frac{z-1}{z+1}\right)+a_n}{b_0\left(\frac{2}{T}\right)^n\left(\frac{z-1}{z+1}\right)^n+b_1\left(\frac{2}{T}\right)^{n-1}\left(\frac{z-1}{z+1}\right)^{n-1}+\cdots+b_{n-1}\frac{2}{T}\left(\frac{z-1}{z+1}\right)+b_n}$$

将其分子、分母同乘以 $(z+1)^n$，可得

$$G_0(z)=\frac{a_0\left(\frac{2}{T}\right)^n(z-1)^n+a_1\left(\frac{2}{T}\right)^{n-1}(z-1)^{n-1}(z+1)+\cdots+a_n(z+1)^n}{b_0\left(\frac{2}{T}\right)^n(z-1)^n+b_1\left(\frac{2}{T}\right)^{n-1}(z-1)^{n-1}(z+1)+\cdots+b_n(z+1)^n}=\frac{A(z)}{B(z)}$$

将 $A(z),B(z)$ 写成向量形式

$$A(z)=(a_n,a_{n-1},\cdots,a_1,a_0)\begin{bmatrix}\left(\frac{2}{T}\right)^0 & & & 0\\ & \left(\frac{2}{T}\right)^1 & & \\ & & \ddots & \\ & & & \left(\frac{2}{T}\right)^{n-1}\\ 0 & & & & \left(\frac{2}{T}\right)^n\end{bmatrix}\begin{bmatrix}(z+1)^n\\ (z+1)^{n-1}(z-1)\\ \vdots\\ (z-1)^n\end{bmatrix}$$

$$B(z)=(b_n,b_{n-1},\cdots,b_1,b_0)\begin{bmatrix}\left(\frac{2}{T}\right)^0 & & & 0\\ & \left(\frac{2}{T}\right)^1 & & \\ & & \ddots & \\ & & & \left(\frac{2}{T}\right)^{n-1}\\ 0 & & & \left(\frac{2}{T}\right)^n\end{bmatrix}\begin{bmatrix}(z+1)^n\\ (z+1)^{n-1}(z-1)\\ \vdots\\ (z-1)^n\end{bmatrix}$$

由于$(z+1)^n,(z+1)^{n-1}(z-1),\cdots,(z-1)^n$均为$n$阶多项式，可得到

$$\begin{bmatrix}(z+1)^n\\ (z+1)^{n-1}(z-1)\\ \vdots\\ (z+1)^{n-i}(z-1)^i\\ \vdots\\ (z-1)^n\end{bmatrix}=\begin{bmatrix}x_{00} & x_{01} & \cdots & x_{0n}\\ x_{10} & x_{11} & \cdots & x_{1n}\\ \vdots & \vdots & & \vdots\\ x_{i0} & x_{i1} & \cdots & x_{in}\\ \vdots & \vdots & & \vdots\\ x_{n0} & x_{n1} & \cdots & x_{nn}\end{bmatrix}\begin{bmatrix}z^n\\ z^{n-1}\\ \vdots\\ z\\ \vdots\\ 1\end{bmatrix}$$

矩阵$\{x_{ij}\}$为$(n+1)\times(n+1)$阶；其中第一行诸元素为$(z+1)^n$的展开式的各系数，第一列诸元素为1，阶次n确定后，这些元素均为已知，并可以证明，其余$n\times n$个元素可由下式求得

$$x_{i,j}=x_{i-1,j}-x_{i,j-1}-x_{i-1,j-1}\quad i,j=1,2,\cdots,n \tag{4.58}$$

从而可得

$$A(z)=(a_n\quad a_{n-1}\quad\cdots\quad a_1\quad a_0)\mathrm{diag}\left(1\quad\frac{2}{T}\quad\cdots\quad\left(\frac{2}{T}\right)^n\right)\{x_{ij}\}[z^n\quad z^{n-1}\quad\cdots\quad z\quad 1]^{\mathrm{T}}$$

$$B(z)=(b_n\quad b_{n-1}\quad\cdots\quad b_1\quad b_0)\mathrm{diag}\left(1\quad\frac{2}{T}\quad\cdots\quad\left(\frac{2}{T}\right)^n\right)\{x_{ij}\}[z^n\quad z^{n-1}\quad\cdots\quad z\quad 1]^{\mathrm{T}}$$

这样就得到了$G_0(z)$分子、分母各系数的表达式

$$(d_0\quad d_1\quad\cdots\quad d_n)=(a_n\quad a_{n-1}\quad\cdots\quad a_1\quad a_0)H \tag{4.59}$$

$$(e_0\quad e_1\quad\cdots\quad e_n)=(b_n\quad b_{n-1}\quad\cdots\quad b_1\quad b_0)H \tag{4.60}$$

其中

$$H=\mathrm{diag}\left(1\quad\frac{2}{T}\quad\cdots\quad\left(\frac{2}{T}\right)^n\right)\{x_{ij}\} \tag{4.61}$$

例如，若

$$G_0(s)=\frac{3s^5+2s^4+s^3+3s^2+2s+1}{s^6+2s^5+3s^4+4s^2+3s+2}$$

当取$T=1$时，其z传递函数$G_0(z)$很容易由计算机程序计算得到，即

$$G_0(z)=\frac{153z^6-402z^5+447z^4+100z^3-505z^2+334z-63}{208z^6-668z^5-1\,252z^4-1\,112z^3+600z^2-204z+52}$$

4.3.2 根匹配法

假定被仿真的连续系统的传递函数如式(4.62)所示，即

$$G(s)=\frac{K(s-q_1)(s-q_2)\cdots(s-q_m)}{(s-p_1)(s-p_2)\cdots(s-p_n)}\quad(n\geqslant m)\tag{4.62}$$

众所周知，一个连续系统动态特性决定于描述该系统的传递函数 $G(s)$，具体地说，决定于 $G(s)$ 的增益及零极点分布，即式(4.62)中的 $K,q_1,\cdots,q_m,p_1,\cdots,p_n$。为了对该系统进行仿真，应设法找到一个 $G(z)$，作为频域仿真建模方法之一的“根匹配法”的基本做法就是由 $z=e^{sT}$ 的转换关系，在 z 平面上也一一对应地确定出零、极点的位置 $q_1',\cdots,q_m',p_1',\cdots,p_n'$，然后根据其他特点(比如终值点)来确定 K_z，则可得如式(4.63)所示的脉冲传递函数 $G(z)$

$$G(z)=\frac{K_z(z-q_1')(z-q_2')\cdots(z-q_m')}{(z-p_1')(z-p_2')\cdots(z-p_n')}\tag{4.63}$$

若 $G(s)$ 是稳定的，即它的全部极点都位于 s 平面的左半平面上，也就是说 $p_1,p_2,\cdots,p_n$ 都具有负实部，那么由式(4.63)所得的 $G(z)$ 也必定稳定。这是因为，z 平面上的极点 $e^{p_1'T}$，$e^{p_2'T},\cdots,e^{p_n'T}$ 都在单位圆内。因此，当采用根匹配法建模时，只要原系统是稳定的，则不论 T 取多大，都能保证仿真模型也是稳定的。根匹配法的一般步骤如下：

(1) 由 $G(s)$ 计算出 $K,q_1,\cdots,q_m,p_1,\cdots,p_n$。

(2) 把 s 平面上的零极点映射到 z 平面上，即 $p_i'=e^{p_iT},q_j'=e^{q_jT},(i=1,2,\cdots,n;\ j=1,2,\cdots,m)$。

(3) 初步构造一个具有上述零极点的 $G(z)$。

(4) 确定 z 平面上的附加零点。因为 $m\leqslant n$，故在 z 平面上有 $n-m$ 个零点在负无穷远处，不妨假设均在 $-\infty$ 处，由此可见，在 z 平面上尚有 $n-m$ 个零点在 $e^{-\infty T}$ 处，即尚有 $n-m$ 个零点在 z 平面的原点。

(5) 在典型输入下，根据终值定理求出连续系统 $G(s)$ 的终值及离散系统 $G(z)$ 的终值。

(6) 根据终值相等的原则确定 K_z。

下面通过一个例子，对根匹配建模方法作进一步分析。设

$$G(s)=\frac{s}{s^2+2s+1}$$

试利用根匹配法，求与之相匹配的离散化模型，步骤如下：

(1) $G(s)=\dfrac{s}{(s+1)^2}$，故 $p_1=-1,p_2=-1,q_1=0,n=2,m=1$；

(2) $p_1'=e^{-T},p_2'=e^{-T},q_1'=1$；

(3) $G(z)=\dfrac{K_z(z-1)}{(z-e^{-T})^2}$；

(4) 附加一个零点，为简单起见，令 $q_2'=0$，即零点配在 z 平面的原点。故

$$G(z)=K_z\frac{z(z-1)}{(z-e^{-T})^2}$$

附加这个零点不影响 K_z；

(5) 加斜坡函数 $u(t)=t$，$G(s)$ 具有非零且有限的稳态值

$$y(\infty)=\lim_{s\to0}[sG(s)U(s)]=\lim_{s\to0}\left[s\frac{s}{(s+1)^2}\frac{1}{s^2}\right]=1$$

由 $G(z)$ 可得

$$y(\infty)=\lim_{z\to1}\left[\frac{z-1}{z}G(z)U(z)\right]$$

$$= \lim_{z\to 1}\left[\frac{z-1}{z}\frac{K_z(z-1)}{(z-e^{-T})^2}\frac{Tz}{(z-1)^2}\right] = \frac{K_z T}{(1-e^{-T})^2} = 1$$

(6) 故 $K_z = \dfrac{(1-e^{-T})^2}{T}$。

$G(s)$的幅频特性与相频特性如表 4.5 中第一行所示。若选 $T=1$，则 $K_z=0.4$，$G(z)$的幅频特性与相频特性如表 4.5 中第二行所示。

比较表 4.5 中 $G(s)$与零点配在原点的 $G(z)$的特性可知，$G(z)$在相位上超前较大，如 $\omega_s=1$ 时，$G(s)$相位为 0°，但 $G(z)$却为 19°左右。

另一种方法是将$(n-m)$个零点配置在 $z=-1$ 处，即在 $G(z)$的分子上乘上$(z+1)^{n-m}$。这样就可得

$$G(z) = \frac{K_z(z-1)(z+1)}{(z-e^{-T})^2} = \frac{K_z(z^2-1)}{(z-e^{-T})^2}$$

用前述相同的方法可得

$$K_z = \frac{(1-e^{-T})^2}{2T}$$

若 $T=0$，则 $K_z=0.2$，该 $G(z)$的幅频特性与相频特性也列于表 4.5 中。由表 4.5 可知，此时相位有一些滞后，如 $\omega_s=1$ 时，$G(s)$相位为 0°，但 $G(z)$却为−6°左右。

为了使相位既不超前也不滞后，可以将一个附加零点设置在(0，−1)之间，即在 $G(z)$的分子上乘上$(z+\delta)^{n-m}$，其中 δ 为(0，1)之间的一个数。即

$$G(z) = \frac{K_z(z-1)(z+\delta)}{(z-e^{-T})^2}$$

现在有两个待定常数：K_z 与 δ，它们可以通过频率特性来确定。即先由 $G(s)$求出 $G(j\omega)$，当 $\omega=1$ 时，可得幅频 $A(\omega)=\dfrac{1}{2}$，相频 $\varphi(\omega)=0°$。

同时，由 $G(z)$也可以求出 $G(j\omega)$。对一定的采样间隔 T，$G(j\omega)$也仅仅是 ω 的函数。为了使两者匹配，应使 $\omega=1$，$A(\omega)$也为 1/2，$\varphi(\omega)$也为 0°，由此可以定出 K_z 及 δ。

设例中 $K_z=0.2828$，$\delta=0.5272$，由此得

$$G(z) = \frac{0.2828(z-1)(z+0.5272)}{(z-0.3679)^2}$$

它的频率特性也列在表 4.5 中，从表中可知不仅幅频误差很小，相频误差也很小。

表 4.5 几种不同的根匹配法的比较

	ω(rad/s)	0.1	0.3	0.6	0.8	1.0	1.2	2.0
幅频特性	连续系统	0.099 01	0.275 2	0.441 2	0.487 8	0.500	0.491 8	0.400 0
	零点配在原点	0.095 37	0.265 9	0.430 7	0.481 2	0.500	0.500 1	0.449 0
	零点配在−1 处	0.108 50	0.299 6	0.468 8	0.505 0	0.500	0.470 0	0.276 5
	零点配在(−1，0)间	0.107 00	0.295 5	0.464 0	0.501 0	0.500	0.474 0	0.302 4
相频特性(°)	连续系统	78.58	56.60	28.070	12.680	0.000	−10.39	−36.87
	零点配在原点	80.50	62.37	39.060	27.990	19.100	12.450	0.350 0
	零点配在−1 处	77.64	53.78	22.410	5.080	−6.020	−6.550	−11.17
	零点配在(−1，0)间	78.53	56.45	27.880	12.530	0.050	−9.960	−31.26

4.3.3 可调整积分法

1. 可调整积分法基本原理

假定有一个连续系统如图 4.16(a)所示。在连续的输入信号 $u(t)$后面再加一个以 T 为周期的采样开关,得一离散信号 $u^*(t)$,然后再加一个信号重构器(其传递函数为 $G_h(s)$),使离散信号 $u^*(t)$再恢复为连续信号;最后将$\tilde{u}(t)$加到原来的连续系统上,其输出为$\tilde{y}(t)$。上述过程如图 4.16(b)所示。显然,只要$\tilde{u}(t)$能足够精确地表示 $u(t)$,那么$\tilde{y}(t)$也就能足够精确地表示 $y(t)$,故图 4.16(b)所示的离散模型也就能足够精确地表示图 4.16(a)所示的连续模型。

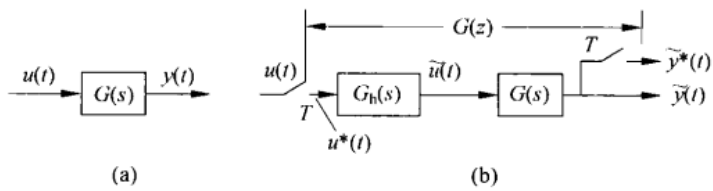


图 4.16 频域离散相似法

由 z 变换的理论可知,从 $u^*(t)$到$\tilde{y}^*(t)$之间离散传递函数 $G(z)$可以用下式来表示

$$G(z) = \mathcal{Z}\{G_h(s)G(s)\} \tag{4.64}$$

其中$\mathcal{Z}$表示对括号中的传递函数求 z 变换。比如,若

$$G_h(s)G(s) = \frac{1}{s(s+a)}$$

则

$$G(z) = \mathcal{Z}\left\{\frac{1}{s(s+a)}\right\} = \frac{1}{a}\frac{(1-e^{-aT})z^{-1}}{(1-z^{-1})(1-e^{-aT}z^{-1})}$$

为了便于读者参考,表 4.6 列出了常用的 z 变换表。

表 4.6 z 变换表

$f(t)$	$F^*(k)(k=1,2,\cdots)$	$F(s)$	$F(z)$
$\delta()$	$[1,0,0,\cdots]$	1	1
$1(t)$	$[1,1,1,\cdots]$	$\frac{1}{s}$	$\frac{1}{1-z^{-1}}$
t	$[0,T,2T,\cdots]$	$\frac{1}{s^2}$	$\frac{Tz^{-1}}{(1-z^{-1})^2}$
e^{-aT}	e^{-aT}	$\frac{1}{s+a}$	$\frac{1}{1-e^{-aT}z^{-1}}$
$\frac{1}{a}(1-e^{-aT})$	$\frac{1}{a}(1-e^{-aT})$	$\frac{1}{s(s+a)}$	$\frac{1}{a}\frac{(1-e^{-aT})z^{-1}}{(1-z^{-1})(1-e^{-aT}z^{-1})}$

离散相似方法具有物理意义明确、程序简单、计算量小的特点,是一种恒稳的计算方法。由于离散相似模型中引入了虚拟采样开关和虚拟信号重构器,所以连续信号离散化为离散信号,然后经重构器恢复成连续信号。从理论上讲,信号重构器应能无失真地将离散信号重

新恢复成原连续信号，但实际上这是非常困难的。如何保证离散模型达到规定精度要求的相似性，是这类方法要解决的基本问题。

2. 信号重构器的频谱特性分析

本节将从频域角度讨论对虚拟采样器的采样周期 T 及信号重构器（也称保持器）的要求。

1）采样周期 T

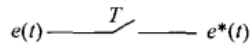


图 4.17 采样开关

假定有一连续信号 $e(t)$，若将它通过一只采样周期为 T 的采样开关（图 4.17），得离散 $e^*(t)$。已知

$$e^*(t)=\frac{1}{T}\sum_{n=-\infty}^{\infty}e(t)\mathrm{e}^{\mathrm{j}n\omega_s t} \tag{4.65}$$

其中，$\omega_s=2\pi/T$，称为采样频率。

对式(4.65)两边取拉氏变换，可得

$$E^*(s)=\frac{1}{T}\sum_{n=-\infty}^{\infty}\mathcal{L}[e(t)\mathrm{e}^{\mathrm{j}n\omega_s t}]$$

根据拉氏变换的平移原理，上式可变为

$$E^*(s)=\frac{1}{T}\sum_{n=-\infty}^{\infty}E[s-\mathrm{j}n\omega_s]$$

用 $\mathrm{j}\omega$ 代替 s，可得

$$E^*(\mathrm{j}\omega)=\frac{1}{T}\sum_{n=-\infty}^{\infty}E[\mathrm{j}(\omega-n\omega_s)] \tag{4.66}$$

若 $E(\mathrm{j}\omega)$ 为 $e(t)$ 的频谱，如图 4.18(a)所示，则离散信号 $e^*(t)$ 的频谱 $E^*(\mathrm{j}\omega)$ 为一周期性频谱，其周期为 ω_s，如图 4.18(b)所示。

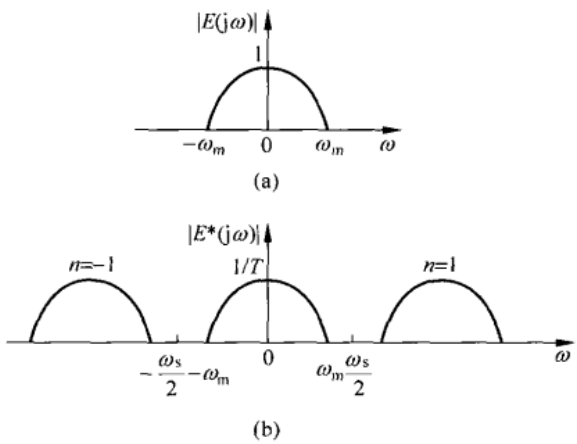


图 4.18 连续信号与离散信号的频谱

由图 4.18 可知，如果 $\omega_s/2\geqslant\omega_m$，那么离散信号的频谱中的基本部分（$n=0$）将与原连续信号的频谱相同（只是幅度上差 $1/T$），也就是若 $\omega_s/2\geqslant\omega_m$，离散信号将全部包含原连续信号的信息。若 $\omega_s/2<\omega_m$，那么 $n=0$ 的部分频谱将和 $n=1$ 及 $n=-1$ 之间的频谱部分重叠，因而使离散频谱的基本部分不再与原连续信号的频谱相同，这种现象称为混叠。此时，离散

信号将不再包含原连续信号的全部信息。

总之，为使离散相似模型与连续模型相似，首先要求采样开关的频率 $\omega_s=2\pi/T$ 大于信号最大频率 ω_m 的两倍。换句话说，采样周期 T 不能过大。

这就是著名的香农采样定理：对于一个具有有限频谱的连续信号（$-\omega_m<\omega<\omega_m$）进行采样，只要选择 $\omega_s/2\geqslant\omega_m$，通过理想的低通滤波器，就能把原来信号毫无失真地提取出来，其中 ω_m 是信号频谱的最大频率，$\omega_s=2\pi/T$，T 是采样周期。采样定理规定的采样频率 $\omega_s/2\geqslant\omega_m$ 这个条件，也就是相邻两频谱互不重叠的条件。

2）理想的信号重构器与实际的信号重构器

根据图 4.18，若 $\omega_s/2\geqslant\omega_m$，则离散信号的频谱没有重叠现象。要求用一个信号重构器完全恢复连续信号，故该信号重构器的频率特性如图 4.19 所示（图中只画出了幅频，相频要求为零度），即要求它有一个锐截止的频率特性。

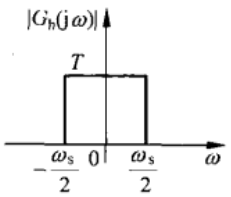


图 4.19 理想信号重构器的频率特性

显然，这种理想的信号重构器是无法实现的，实际上能实现的只是各种近似于图 4.19 特性的信号重构器。下面介绍几种实际的信号重构器。

（1）零阶信号重构器

它是将离散信号在两个采样点之间保持不变，因而使离散信号恢复为一个阶梯状的连续信号，如图 4.20 所示。

这个零阶信号重构器的脉冲过渡函数如图 4.21 所示，故可计算出它的传递函数为

$$G_h(s)=\mathcal{L}[g_h(t)]=\frac{1-e^{-Ts}}{s} \tag{4.67}$$

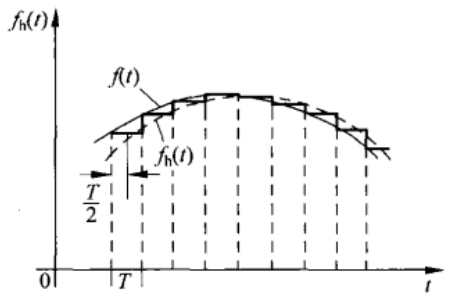


图 4.20 通过零阶信号重构器后的信号

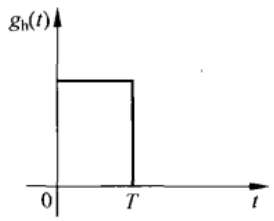


图 4.21 零阶信号重构器的脉冲过渡函数

根据式(4.67)，可画出它的频率特性如图 4.22 所示。由图 4.22 可知，它与理想的信号重构器相比（见图 4.19），在幅频上略有误差，而相频上则略有延迟。由图 4.20 也可以看出，将 $f_h(t)$ 光滑处理，它大约比 $f(t)$ 延迟 $T/2$。需要指出的是，若输入信号为阶跃信号，那么零阶信号重构器能无失真地恢复信号。

（2）一阶信号重构器

一阶信号重构器是以当前时刻和前一时刻两个采样值为基础进行外推，也即要求 $f_h(t)$ 满足

$$f_h(t)=f(kT)+\frac{f(kT)-f[(k-1)T]}{T}(t-kT),\quad kT\leqslant t\leqslant(k+1)T \tag{4.68}$$

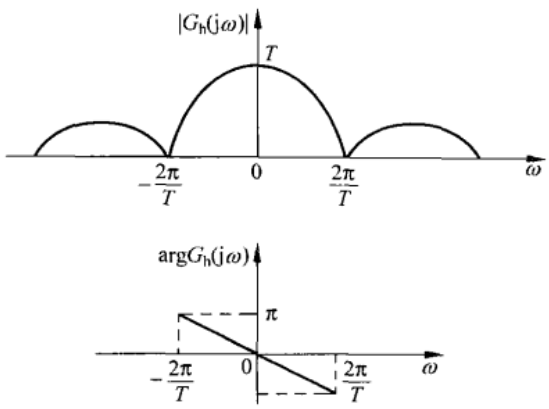


图 4.22 零阶信号重构器的频谱特性

它的脉冲过渡函数及频率特性如图 4.23 及图 4.24 所示，传递函数为式(4.69)。

$$G_h(s) = \mathcal{L}[g_h(t)] = T(1+Ts)\left(\frac{1-e^{-Ts}}{Ts}\right)^2 \qquad (4.69)$$

由图 4.24 可知，一阶信号重构器的幅频也有误差，相频也有延迟。这种信号重构器只能无失真地恢复斜坡信号。对于一般信号，它恢复的情况如图 4.25 所示。

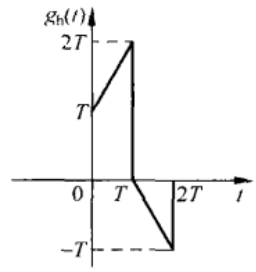


图 4.23 一阶信号重构器的脉冲过渡函数

(3) 三角形信号重构器

上述两种信号重构器对一般信号都有较大的误差，为了减少误差，可以采用三角形信号重构器，它满足

$$f_h(t) = f(kT) + \frac{f[(k+1)T] - f(kT)}{T}, \quad kT \leqslant t \leqslant (k+1)T \qquad (4.70)$$

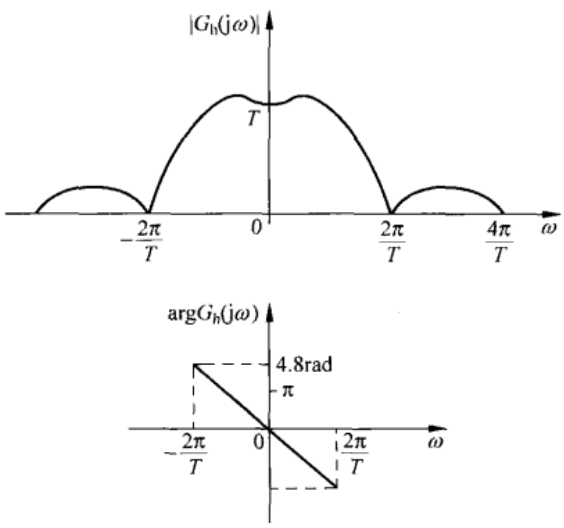


图 4.24 一阶信号重构器的频率特性

它的脉冲过渡函数如图 4.26 所示，由于形状为三角形，因而得此名。它的传递函数为

$$G_h(s) = \mathcal{L}[g_h(t)] = \frac{e^{Ts}}{T}\left(\frac{1-e^{-sT}}{Ts}\right)^2 \tag{4.71}$$

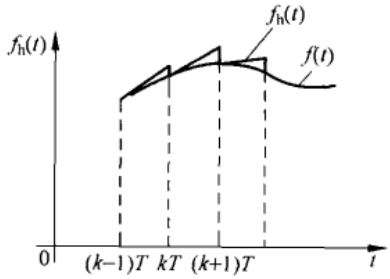


图 4.25 一阶信号重构器的恢复特性

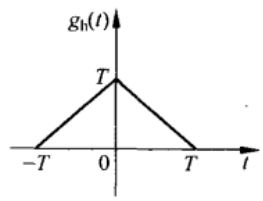


图 4.26 三角形信号重构器的脉冲过渡函数

它的频率特性如图 4.27 所示。由于它相对纵轴左右对称，故只画了右半边；另外，由于相频保持为零度，故图中未画出来。

由图 4.27 可知，三角形信号重构器高频部分失真很小，且无相位滞后，所以可以获得比较满意的结果，它恢复的 $f_h(t)$ 如图 4.28 所示。

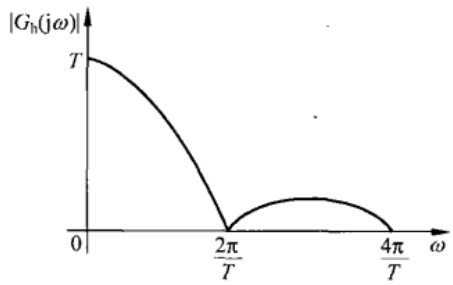


图 4.27 三角形信号重构器的频率特性

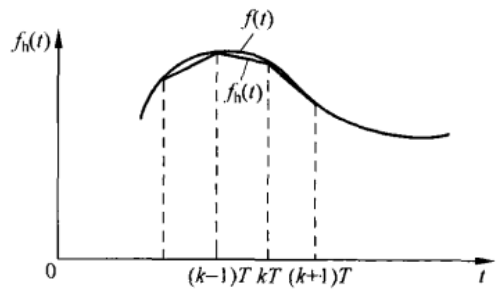


图 4.28 三角形信号重构器的恢复特性

虽然三角形信号重构器有较好的特性，但是这种信号重构器要求在计算 $(k+1)T$ 的 $f_h(t)$ 时已知 $f[(k+1)T]$，有时这是不可能的，所以还有一种滞后一拍的三角形信号重构器可以被采用，它满足下式

$$f_h(t) = f[(k-1)T] + \frac{f(kT) - f[(k-1)T]}{T}(t-kT),$$
$$kT \leqslant t \leqslant (k+1)T \tag{4.72}$$

它的 $g_h(t)$ 及 $G_h(j\omega)$ 如图 4.29 所示。其数学表达式为

$$G_h(s) = \left(\frac{1-e^{-sT}}{Ts^2}\right)^2 \tag{4.73}$$

由以上分析可见，这四种信号重构器都不是理想的低通滤波器，即幅度随频率提高而减少；截止频率是 n 个，即除了允许主要频率分量通过外，还允许高频频谱分量通过，因此，被重构的信号会失真。与其他高阶信号重构器相比较，零阶信号重构

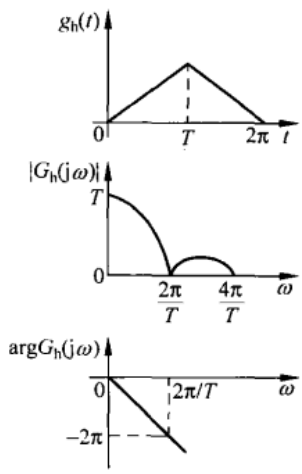


图 4.29 滞后一拍的三角形信号重构器

器简单、容易实现，并具有较小相位滞后，所以在实际系统中经常被采用。

3）可调整积分法

由上一节的讨论我们知道，频域离散相似仿真方法是在连续系统中加上虚拟采样开关和信号重构器。若选择 $\omega_s \geqslant 2\omega_m$，则离散信号包含原连续信号的全部信息，因而有可能重构连续信号。然而，由于实际信号往往具有较宽的频谱，采样频率难以完全满足采样定理的要求，又加上实际信号重构器不可能无失真地重构信号，因此必然引入仿真误差，具体体现在信号的相位延迟和幅度衰减。为了减少离散相似模型的误差，有时需要校正，也就是在模型中加入一个补偿器，适当调整其相位、幅度，以便尽可能接近原连续信号。补偿方法有连续补偿和离散补偿两种。

（1）连续补偿

在信号重构器后加一个连续型补偿器，如图 4.30 所示。连续型补偿器可以采用如下形式

$$G_{补}(s) = \lambda e^{\gamma T s} \tag{4.74}$$

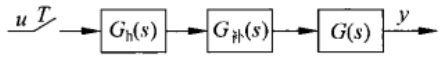


图 4.30 连续补偿

其中 λ 称为幅度补偿因子，用来进行幅度上的补偿；γ 则用来进行相位上的补偿，称为相位补偿因子。适当调整 λ 和 γ 可以使离散模型尽可能接近原型。

式(4.74)是超越函数，为方便实际使用，往往将其用有理式近似。常用的连续补偿器形式如表 4.7 所示。

表 4.7 常用的三种连续型补偿器的形式

类 型	$G_{补}(s)$	导 出 式
单零点补偿	$1+\gamma Ts$	$e^{\gamma Ts}=1+\gamma Ts+\frac{\gamma^2}{2}T^2s^2+\cdots\approx 1+\gamma Ts$
单极点补偿	$\frac{1}{1-\gamma Ts}$	$e^{\gamma Ts}=\frac{1}{e^{-\gamma Ts}}=\frac{1}{1-\gamma Ts+\frac{\gamma^2}{2}T^2s^2+\cdots}\approx\frac{1}{1-\gamma Ts}$
单零极点补偿	$\frac{2+\gamma Ts}{2-\gamma Ts}$	$e^{\gamma Ts}=\frac{e^{\gamma Ts/2}}{e^{-\gamma Ts/2}}=\frac{1+\frac{1}{2}\gamma Ts+\frac{1}{8}\gamma^2T^2s^2+\cdots}{1-\frac{1}{2}\gamma Ts+\frac{1}{8}\gamma^2T^2s^2+\cdots}\approx\frac{2+\gamma Ts}{2-\gamma Ts}$

（2）离散补偿

在信号重构器前加一个离散型补偿器，如图 4.31 所示。



图 4.31 离散补偿

这一般用于实际的采样控制系统中，因为重构器前的采样开关是实际的，譬如说是计算机，这样补偿器往往就放在计算机里，在逻辑上相当于放在重构器的前面。

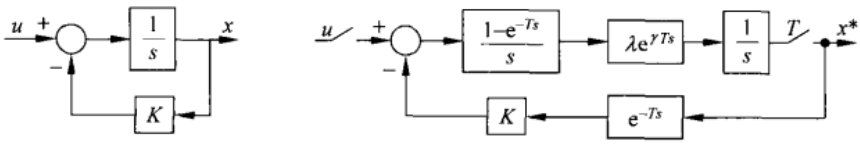
下面举例说明如何选择补偿器的参数。

例 4.2 已知原系统的框图如图 4.32(a)所示，它的等价离散系统如图 4.32(b)所示。

图中，对输入信号 u 及反馈信号均进行了离散化，然后加零阶信号重构器。为补偿信号重构器的误差，加一个连续型补偿器 $\lambda e^{\gamma Ts}$。

至于反馈回路中的 e^{-Ts}，是考虑到在计算反馈回路之前，必须先计算正向回路，因此将反馈回路延迟了一拍。不难看出，原系统模型为

$$\frac{x(s)}{u(s)} = \frac{1}{s+K} \tag{4.75}$$



(a) 补偿器参数选择举例　　(b) 连续系统与等价的离散系统

图 4.32 例 4.2 图

若选择单零点补偿方式，则

$$\lambda e^{\gamma Ts} \approx \lambda(1+\gamma Ts)$$

对图 4.32(b)进行 z 变换，可得

$$\frac{x(z)}{u(z)} = \frac{\lambda Tz[\gamma z + (1-\gamma)]}{z^2 + (\lambda KT\gamma - 1)z + K\lambda T(1-\gamma)} \tag{4.76}$$

它的特征根为

$$z_{极点} = \frac{1 - KT\gamma\lambda}{2} \pm \frac{1}{2}[(\lambda KT\gamma - 1)^2 - 4KT\lambda(1-\gamma)]^{\frac{1}{2}}$$

为了使图 4.32(b)与(a)等价，首先系统阶次要相同，选 γ 与 λ 使 z 平面上的两个 z 极点变成一个，同时使它与原连续系统的极点 $s_{极点} = -K$ 相匹配。由式(4.76)可知，可选择 $\gamma=1$，有

$$\frac{X(z)}{U(z)} = \frac{\lambda Tz}{z + (K\lambda T - 1)} \tag{4.77}$$

此时它只有一个极点：$z_{极点} = 1 - K\lambda T$，为使它与 $s_{极点} = -K$ 相匹配，即要求

$$z_{极点} = e^{-KT} = 1 - K\lambda T$$

由此可得

$$\lambda = \frac{1 - e^{-KT}}{KT}$$

将 $r=1$ 及 $\lambda = \dfrac{1-e^{-KT}}{KT}$ 代入式(4.76)，得

$$\frac{X(z)}{U(z)} = \frac{z(1 - e^{-KT})}{K(z - e^{-KT})} \tag{4.78}$$

■

一般来说，在确定 λ，γ 后，应按照根匹配法的要求，对整个系统的稳态增益系数进行调整，即选择一作用函数，使离散系统与原连续系统具有相同的、有限但不为零的稳态值。

原系统为“0”型，可选单位阶跃作用函数，其稳态增益为

$$x_{\infty} = \lim_{s \to 0} \frac{1}{s} \frac{1}{s+K} s = \frac{1}{K} \tag{4.79}$$

则离散化后系统的稳态增益为

$$x_\infty = \lim_{z\to 1}\frac{1}{z-1}\frac{z}{K}\frac{1-\mathrm{e}^{-KT}}{z-\mathrm{e}^{-KT}}(z-1)K_z = \frac{1}{K} \tag{4.80}$$

其中 K_z 称为稳态增益调整因子，本系统恰好为 1。式(4.78)所对应的差分方程为

$$x(k+1) = \mathrm{e}^{-KT}x(k) + \frac{1}{K}(1-\mathrm{e}^{-KT})u(k+1) \tag{4.81}$$

这就是经过校正后的离散模型。

显然，选择不同的 λ 和 γ 可以得到不同精度的差分模型，这种方法也称“可调整积分法”。

4.4 离散相似法应用举例

本节以大型雷达天线控制系统的仿真设计为例，讨论离散相似法的应用，作为本章内容的总结。

雷达是经典的通信设备，已经广泛应用于军事、气象、航空、航天、环境监测等领域。随着科学技术的不断发展，雷达系统的精度要求日益提高。为了提高雷达系统的精度，天线的尺寸也越来越大。天线尺寸的加大引发了一系列新问题。首先，雷达伺服系统中的传动机构不能视其为刚体链接，必须考虑其弹性变形，因为大型天线的巨大转动惯量使得传动机构的弹性变形难以忽略，从而传动机构具有一定的谐振特性。转动惯量越大，谐振频率越低，人们称为机械谐振，这给提高伺服系统的带宽带来很大困难；同时，风扰力矩的影响随天线尺寸增大而迅速加大，如果伺服系统的带宽不够，可能会影响雷达的跟踪精度。因此，如何克服大型雷达的机械谐振是大型雷达控制系统设计必须解决的问题。

4.4.1 机械谐振分析

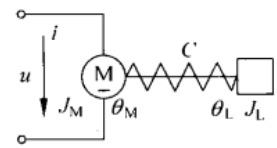


图 4.33 单电机驱动的雷达伺服系统示意图

下面，我们进一步分析这一问题。为便于讨论，先考虑单电机驱动。单电机驱动的雷达伺服系统的原理示意如图 4.33，其中，天线转动惯量及传动机构的弹性系数均已折算到电机轴上。图 4.33 及后面方程中用到的符号说明如表 4.8 所示。

表 4.8 符号说明

符号	含 义	符号	含 义
u	电机电枢端电压	θ_M	电机轴转角
i	电机电枢回路电流	θ_L	天线转角
R	电机电枢回路电阻	M_c	风扰力矩
C_e	电机电势系数	f_L	天线等效黏性摩擦系数
C_m	电机转矩系数	J_L	天线等效转动惯量
J_M	电机转动惯量	C	传动机构弹性系数
f_M	电机等效黏性摩擦系数		

该系统的模型为

$$\begin{cases} u = Ri + C_e s\theta_M \text{（未考虑电枢回路电感）} \\ J_M s^2\theta_M = C(\theta_L - \theta_M) + C_M i - f_M s\theta_M \\ J_L s^2\theta_L = C(\theta_M - \theta_L) - f_L s\theta_L + M_c \end{cases} \tag{4.82}$$

其结构图如图 4.34 所示。

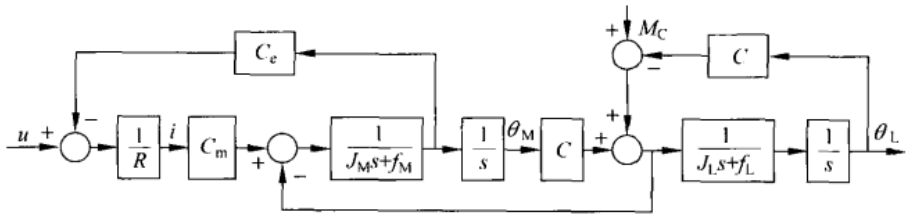


图 4.34 单电机驱动的雷达伺服系统结构图

从式(4.82)或图 4.34，可得其传递函数为

$$W_1(s) = \frac{\theta_M}{u}$$

$$= \frac{\frac{C_m}{R}\left(\frac{J_L}{C}s^2 + \frac{f_L}{C}s + 1\right)}{s\left[\frac{J_M J_L}{C}s^3 + \left(\frac{J_M f_L}{C} + \frac{J_L C_m C_e}{RC}\right)s^2 + \left(J_M + J_L + \frac{C_m C_e f_L}{RC} + \frac{f_M f_L}{C}\right)s + \frac{C_m C_e}{R} + f_M + f_L\right]}$$

$$W_2(s) = \frac{\theta_L}{u}$$

$$= \frac{\frac{C_m}{R}}{s\left[\frac{J_M J_L}{C}s^3 + \left(\frac{J_M f_L}{C} + \frac{J_L C_m C_e}{RC}\right)s^2 + \left(J_M + J_L + \frac{C_m C_e f_L}{RC} + \frac{f_M f_L}{C}\right)s + \frac{C_m C_e}{R} + f_M + f_L\right]}$$

当天线转动惯量较小时，可以忽略传动机构的弹性变形，即认为是刚性连接($C\to\infty$)，则

$$W_2(s) = W_1(s) = \frac{\theta_M}{u} = \frac{\theta_L}{u} = \frac{C_m/R}{(J_M + J_L)s + C_e C_m/R + f_M + f_L} \cdot \frac{1}{s} = \frac{K_0}{T_0 s + 1} \cdot \frac{1}{s} \tag{4.83}$$

若弹性变形不能忽略，那么 C 为一有限值，则

$$W_2(s) = \frac{\theta_L}{u} = \frac{K_0}{(T_1 s + 1)(T_2^2 s^2 + 2T_2\zeta s + 1)} \cdot \frac{1}{s} \tag{4.84}$$

其中，$T_0 = R(J_M + J_L)/(C_e C_m + R(f_M + f_L))$，$K_0 = C_m/(C_e C_m + R(f_M + f_L))$，$T_0 = T_1 T_2^2$。比较式(4.83)和式(4.84)，不难看出，后者增加了一个二阶环节。而且，一般说来，该环节的阻尼很小，因而是振荡环节。这种由于传动机构的弹性变形而产生的振荡称为机械谐振，而 $\omega = 1/T_2$ 称为机械谐振频率，其值主要决定于天线转动惯量及传动机构的弹性系数。

正是这种机械谐振，大大影响了系统的稳定性。图 4.35(a)是当 $C\to\infty$ 时 $W_2(j\omega)$ 的幅频特性。图 4.35(b)是 C 为有限值时 $W_2(j\omega)$ 的幅频特性，在 $\omega = 1/T_2$ 有一谐振点。为保证系统稳定，在设计控制器时要减小开环增益，结果其开环剪切频率 ω_c 大大降低了(图 4.35(c))，从而影响了系统的快速跟踪性能。

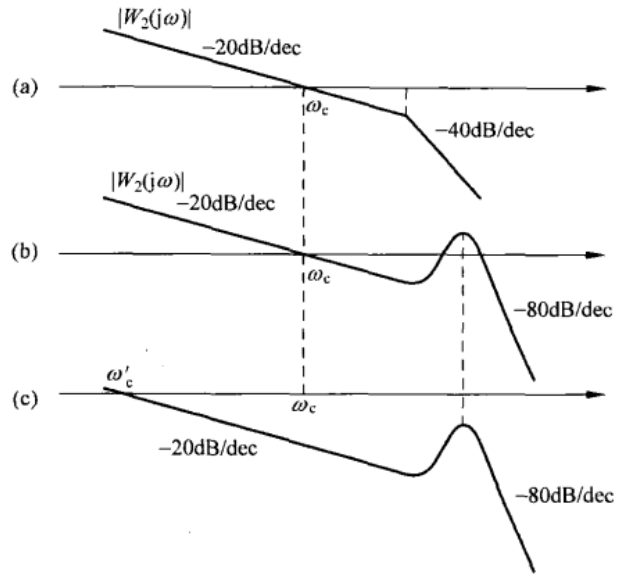


图 4.35 机械谐振频率特性示意

4.4.2 模型设计

仿真设计的第一步是模型设计，必须建立雷达伺服系统及其控制系统的模型。为此首先介绍克服机械谐振的设计方案，从而才能确定控制系统的模型。

实践表明，按传统双环调节（电流调节、速度调节）设计方法是很难克服机械谐振的，只有采用全状态反馈，按天线跟踪的频带要求配置系统的极点，增加引起机械谐振的一对极点的阻尼，才能有效地克服机械谐振。显然要确定系统的全状态反馈系数，单靠工程技术人员的经验试凑是不可能达到的，必须借助计算机仿真这一工具。

下面先简单讨论大型雷达控制系统全状态反馈设计方法。

（1）单电机驱动的雷达伺服系统

对于图 4.33 所示的单电机驱动的雷达伺服系统，设计时还必须考虑电机电枢回路的电感（L），系统用状态方程描述如下

$$\begin{bmatrix}\dot{\theta}_M\\ \dot{\omega}_M\\ \dot{i}\\ \dot{\omega}_L\\ \dot{\theta}_L\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0 & 0\\ -\dfrac{C}{J_M} & -\dfrac{f_M}{J_M} & \dfrac{C_m}{J_M} & 0 & \dfrac{C}{J_M}\\ 0 & -\dfrac{C_e}{L} & -\dfrac{R}{L} & 0 & 0\\ \dfrac{C}{J_L} & 0 & 0 & -\dfrac{f_L}{J_L} & -\dfrac{C}{J_L}\\ 0 & 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\theta_M\\ \omega_M\\ i\\ \omega_L\\ \theta_L\end{bmatrix}+\begin{bmatrix}0 & 0\\ 0 & 0\\ \dfrac{1}{L} & 0\\ 0 & \dfrac{1}{J_L}\\ 0 & 0\end{bmatrix}\begin{bmatrix}u\\ M_c\end{bmatrix} \tag{4.85}$$

式中的符号除了前面已经说明的之外，$\omega_M=\mathrm{d}\theta_M/\mathrm{d}t$ 为电机转速，$\omega_L=\mathrm{d}\theta_L/\mathrm{d}t$ 为天线折算到电机轴上的转速。

如前所述，该系统对机械谐振的阻尼很小，需要采用全状态反馈配置极点以加大机械谐

振的阻尼。但是，由于式(4.85)的状态变量中含有 θ_M，如果引进 θ_M 的反馈势必降低系统的静差度，这是随动系统所不希望的。因此需要找到一个新的状态变量代替 θ_M 的反馈而又不会影响系统的静差度。

考虑到 θ_M 与 θ_L 是通过传动机构的弹性变形产生的弹性力矩作用联系的，而弹性力矩与天线角加速度有关，若采用天线角加速度反馈，则可增加系统的阻尼而不会影响系统的静差度。

记 $\varepsilon=\mathrm{d}\omega_L/\mathrm{d}t$，代入式(4.85)得到

$$\theta_M=\frac{J_L}{C}\varepsilon+\frac{f_L}{C}\omega_L-\frac{M_c}{C}+\theta_L$$

又

$$\frac{\mathrm{d}\varepsilon}{\mathrm{d}t}=\frac{\mathrm{d}^2\omega_L}{\mathrm{d}t^2}=\frac{C}{J_L}\omega_M-\frac{C}{J_L}\omega_L-\frac{f_L}{J_L}\varepsilon+\frac{1}{J_L}\dot{M}_c$$

若设 $\dot{M}_c=\mathrm{d}M_c/\mathrm{d}t=0$，将 θ_M，ε，$\mathrm{d}\varepsilon/\mathrm{d}t$ 代入式(4.85)，则可得到

$$\begin{bmatrix}\dot{\varepsilon}\\ \dot{\omega}_M\\ \dot{i}\\ \dot{\omega}_L\\ \dot{\theta}_L\end{bmatrix}=\begin{bmatrix}-\frac{f_L}{J_L} & \frac{C}{J_L} & 0 & -\frac{C}{J_L} & 0\\ \frac{J_L}{J_M} & -\frac{f_M}{J_M} & \frac{C_m}{J_M} & -\frac{f_L}{J_M} & 0\\ 0 & -\frac{C_e}{L} & -\frac{R}{L} & 0 & 0\\ 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\varepsilon\\ \omega_M\\ i\\ \omega_L\\ \theta_L\end{bmatrix}+\begin{bmatrix}0 & 0\\ 0 & \frac{1}{J_M}\\ \frac{1}{L} & 0\\ 0 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}u\\ M_c\end{bmatrix}\tag{4.86}$$

取式(4.86)中五个状态变量反馈，不但可以根据需要配置系统的极点，而且避免 θ_M 的反馈，不会降低系统的静差度。

(2) 双电机驱动的雷达伺服系统

前面讨论了单电机驱动系统，实际上，对大型雷达伺服系统来说，天线尺寸加大后，驱动力矩要求也大大增加，一般单电机难以胜任，并且其传动间隙也很难消除，工程上往往采用双电机驱动。经过必要的简化及标准化后，双电机驱动的雷达伺服系统可用图 4.36 表示。

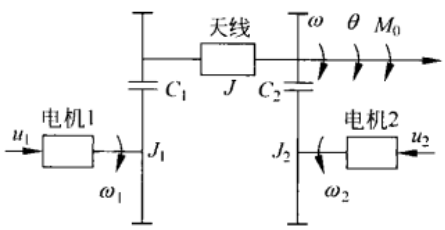


图 4.36 双电机驱动的雷达伺服系统

图 4.36 中的符号意义与单电机驱动时所说明的相同，下标 1、2 分别表示电机 1 和电机 2 及相应的传动机构的参数，无下标的为天线参数。与前类似，可以导出系统的运动方程如下

$$\dot{\omega}_1=-\frac{f_1}{J_1}\omega_1-\frac{C_1}{J_1}\theta_1+\frac{C_{m1}}{J_1}i_1+\frac{C_1}{J_1}\omega\quad\dot{\theta}_1=\omega_1$$

$$\dot{\omega}_2=-\frac{f_2}{J_2}\omega_2-\frac{C_2}{J_2}\theta_2+\frac{C_{m2}}{J_2}i_2+\frac{C_2}{J_2}\omega\quad\dot{\theta}_2=\omega_2$$

$$\dot{i}_1=-\frac{C_{e1}}{L_1}\omega_1-\frac{R_1}{L_1}i_1+\frac{1}{L_1}u_1\quad\dot{\omega}=\frac{C_1}{J}\theta_1+\frac{C_2}{J}\theta_2-\frac{C_1+C_2}{J}\theta-\frac{f}{J}\omega+\frac{1}{J}M_c$$

$$\dot{i}_2=-\frac{C_{e2}}{L_2}\omega_2-\frac{R_2}{L_2}i_2+\frac{1}{L_2}u_2\quad\dot{\theta}=\omega$$

写成状态方程的形式

$$\begin{bmatrix}\dot{\omega}_1\\ \dot{\omega}_2\\ \dot{\theta}_1\\ \dot{\theta}_2\\ \dot{i}_1\\ \dot{i}_2\\ \dot{\theta}\\ \dot{\omega}\end{bmatrix}=\begin{bmatrix}-\frac{f_1}{J_1} & 0 & -\frac{C_1}{J_1} & 0 & \frac{C_{m1}}{J_1} & 0 & 0 & \frac{C_1}{J_1}\\ 0 & -\frac{f_2}{J_2} & 0 & -\frac{C_2}{J_2} & 0 & \frac{C_{m2}}{J_2} & 0 & \frac{C_2}{J_2}\\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ -\frac{C_{e1}}{L_1} & 0 & 0 & 0 & -\frac{R_{e1}}{L_1} & 0 & 0 & 0\\ 0 & -\frac{C_{e2}}{L_2} & 0 & 0 & 0 & -\frac{R_{e2}}{L_2} & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & \frac{C_1}{J} & \frac{C_2}{J} & 0 & 0 & -\frac{C_1+C_2}{J} & -\frac{f}{J}\end{bmatrix}\begin{bmatrix}\omega_1\\ \omega_2\\ \theta_1\\ \theta_2\\ i_1\\ i_2\\ \theta\\ \omega\end{bmatrix}+\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\\ \frac{1}{L_1} & 0 & 0\\ 0 & \frac{1}{L_2} & 0\\ 0 & 0 & 0\\ 0 & 0 & \frac{1}{J}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\\ M_c\end{bmatrix} \tag{4.87}$$

如果两电机参数完全对称，并标以下标 M，传动系统的刚度相同，即

$$f_1=f_2=f_M,\quad C_{e1}=C_{e2}=C_e,\quad C_{m1}=C_{m2}=C_m$$
$$J_1=J_2=J_M,\quad R_1=R_2=R,\quad L_1=L_2=L,\quad C_1=C_2=C$$

现做如下变换

$$\omega_s=\omega_1+\omega_2,\quad \theta_s=\theta_1+\theta_2,\quad i_s=i_1+i_2$$
$$\omega_d=\omega_1-\omega_2,\quad \theta_d=\theta_1-\theta_2,\quad i_d=i_1-i_2$$

称(θ_s,ω_s,i_s)为和速子空间，(θ_d,ω_d,i_d)为差速子空间。这样，式(4.87)可变换为如下形式

$$\begin{bmatrix}\dot{\theta}_d\\ \dot{\omega}_d\\ \dot{i}_d\\ \dot{\theta}_s\\ \dot{\omega}_s\\ \dot{i}_s\\ \dot{\theta}\\ \dot{\omega}\end{bmatrix}=\left[\begin{array}{ccc:ccccc}0 & 1 & 0 & & & & & \\ -\frac{C}{J_M} & -\frac{f_M}{J_M} & \frac{C_m}{J_M} & & & 0 & & \\ 0 & -\frac{C_e}{L} & -\frac{R}{L} & & & & & \\ \hdashline & & & 0 & 1 & 0 & 0 & 0\\ & & & -\frac{C}{J_M} & -\frac{f_M}{J_M} & \frac{C_m}{J_M} & \frac{2C}{J_M} & 0\\ & 0 & & 0 & -\frac{C_e}{L} & -\frac{R}{L} & 0 & 0\\ & & & 0 & 0 & 0 & 0 & 1\\ & & & \frac{C}{J} & 0 & 0 & -\frac{2C}{J} & -\frac{f}{J}\end{array}\right]\begin{bmatrix}\theta_d\\ \omega_d\\ i_d\\ \theta_s\\ \omega_s\\ i_s\\ \theta\\ \omega\end{bmatrix}+\left[\begin{array}{ccc}0 & 0 & 0\\ 0 & 0 & 0\\ \frac{1}{L} & 0 & 0\\ \hdashline 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & \frac{1}{L} & 0\\ 0 & 0 & 0\\ 0 & 0 & \frac{1}{J}\end{array}\right]\begin{bmatrix}u_1-u_2\\ u_1+u_2\\ M_c\end{bmatrix} \tag{4.88}$$

从式(4.88)可以看到，和速子空间与差速子空间是完全解耦的，设计控制器时可以分别考虑。

(3) 和速子空间

先讨论和速子空间，其方程为

$$\begin{bmatrix}\dot{\theta}_s\\ \dot{\omega}_s\\ \dot{i}_s\\ \dot{\omega}\\ \dot{\theta}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0 & 0\\ -\frac{C}{J_M} & -\frac{f_M}{J_M} & \frac{C_m}{J_M} & 0 & \frac{2C}{J_M}\\ 0 & -\frac{C_e}{L} & -\frac{R}{L} & 0 & 0\\ \frac{C}{J} & 0 & 0 & -\frac{f}{J} & -\frac{2C}{J}\\ 0 & 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\theta_s\\ \omega_s\\ i_s\\ \omega\\ \theta\end{bmatrix}+\begin{bmatrix}0 & 0\\ 0 & 0\\ \frac{1}{L} & 0\\ 0 & \frac{1}{J}\\ 0 & 0\end{bmatrix}\begin{bmatrix}u_1+u_2\\ M_c\end{bmatrix} \tag{4.89}$$

比较式(4.89)与式(4.85)两式，可以看出，和速子空间状态方程与单电机驱动的状态方程结构完全相同，只是前者的状态变量 $\omega_s=\omega_1+\omega_2$，$\theta_s=\theta_1+\theta_2$，$i_s=i_1+i_2$ 分别代替了后者的 ω_M,θ_M,i。这说明，在两电机及其传动机构参数完全对称的情况下，单电机驱动的系统与双电机驱动的系统从克服机械谐振的角度来看没有本质区别。

参照单电机驱动的设计思想，为避免 θ_s 反馈，用天线角加速度 $\varepsilon=\mathrm{d}\omega/\mathrm{d}t$ 代替 θ_s 反馈，即做如下变换

$$\theta_s=\frac{J}{C}\varepsilon+\frac{f}{C}\omega_s-\frac{M_c}{C}+2\theta$$

$$\frac{\mathrm{d}\varepsilon}{\mathrm{d}t}=\frac{\mathrm{d}^2\omega}{\mathrm{d}t^2}=\frac{C}{J}\omega_s-\frac{2C}{J}\omega-\frac{f}{J}\varepsilon+\frac{1}{J}\dot{M}_c$$

设 $\dot{M}_c=\mathrm{d}M_c/\mathrm{d}t$，$u_s=u_1+u_2$，则式(4.89)成为如下形式

$$\begin{bmatrix}\dot{\varepsilon}\\ \dot{\omega}_s\\ \dot{i}_s\\ \dot{\omega}\\ \dot{\theta}\end{bmatrix}=\begin{bmatrix}-\frac{f}{J} & \frac{C}{J} & 0 & -\frac{2C}{J} & 0\\ -\frac{J}{J_M} & -\frac{f_M}{J_M} & \frac{C_m}{J_M} & -\frac{f}{J_M} & 0\\ 0 & -\frac{C_e}{L} & -\frac{R}{L} & 0 & 0\\ 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\varepsilon\\ \omega_s\\ i_s\\ \omega\\ \theta\end{bmatrix}+\begin{bmatrix}0 & 0\\ 0 & \frac{1}{J_M}\\ \frac{1}{L} & 0\\ 0 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}u_s\\ M_c\end{bmatrix} \tag{4.90}$$

这样将单电机驱动的系统与双电机驱动的系统的运动方程统一起来，可以采用同一思想进行克服谐振的控制器设计。

分析式(4.90)，其系数矩阵的第 5 列元素全为 0，又考虑到随动系统要对天线转角通过接收机自动实现单位反馈，因此在模型设计时，可不考虑 θ 的作用，从而可进一步简化为

$$\begin{bmatrix}\dot{\varepsilon}\\ \dot{\omega}_s\\ \dot{i}_s\\ \dot{\omega}\end{bmatrix}=\begin{bmatrix}-\frac{f}{J} & \frac{C}{J} & 0 & -\frac{2C}{J}\\ -\frac{J}{J_M} & -\frac{f_M}{J_M} & \frac{C_m}{J_M} & -\frac{f}{J_M}\\ 0 & -\frac{C_e}{L} & -\frac{R}{L} & 0\\ 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\varepsilon\\ \omega_s\\ i_s\\ \omega\end{bmatrix}+\begin{bmatrix}0 & 0\\ 0 & \frac{1}{J_M}\\ \frac{1}{L} & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}u_s\\ M_c\end{bmatrix} \tag{4.91}$$

式(4.91)习惯上称为速度环，其中三个变量 $\omega_s=\omega_1+\omega_2$，$i_s=i_1+i_2$，$\omega$ 是可以直接测量得到的，因而其状态反馈在物理上是可以实现的。只是天线加速度 ε 难以高精度测量，为此必须寻找 ε 反馈物理实现手段。一种可行的办法是采用观测器理论来估计 ε 这一状态分量，例

如，采用 Leunberger 降维观测器。下面做一简单介绍。

(4) 天线加速度 ε 观测器

对一 n 阶系统，若状态变量 $\boldsymbol{x}$ 中，$\boldsymbol{x}_1$ 是不可直接测量的，$\boldsymbol{x}_2$ 是可直接测量的

$$\begin{bmatrix}\dot{\boldsymbol{x}}_1\\ \dot{\boldsymbol{x}}_2\end{bmatrix}=\begin{bmatrix}\boldsymbol{A}_{11} & \boldsymbol{A}_{12}\\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix}+\begin{bmatrix}\boldsymbol{B}_1\\ \boldsymbol{B}_2\end{bmatrix}\boldsymbol{u}$$

$$\boldsymbol{y}=[0 \quad 1][\boldsymbol{x}_1 \quad \boldsymbol{x}_2]^{\mathrm{T}}$$

若对 $\boldsymbol{x}_1$ 构造观测器，$\boldsymbol{x}_1$ 的观测值为 $\hat{\boldsymbol{x}}_1$，则观测器方程为

$$\begin{cases}\dot{\boldsymbol{z}}=(\boldsymbol{A}_{11}-\boldsymbol{G}_1\boldsymbol{A}_{21})\boldsymbol{z}+\widetilde{\boldsymbol{G}}_1\boldsymbol{y}+\widetilde{\boldsymbol{B}}_1\boldsymbol{u}\\ \hat{\boldsymbol{x}}_1=\boldsymbol{z}+\boldsymbol{G}_1\boldsymbol{y}\end{cases}$$

其中 $\boldsymbol{G}_1$ 由设计人员根据系统要求确定，而

$$\widetilde{\boldsymbol{G}}_1=(\boldsymbol{A}_{11}-\boldsymbol{G}_1\boldsymbol{A}_{21})\boldsymbol{G}_1+\boldsymbol{A}_{12}-\boldsymbol{G}_1\boldsymbol{A}_{22}$$

$$\widetilde{\boldsymbol{B}}_1=\boldsymbol{B}_1-\boldsymbol{G}_1\boldsymbol{B}_2$$

对式(4.91)所表示的系统，则有

$$\boldsymbol{x}_1=\varepsilon,\quad \boldsymbol{x}_2=[\omega_s,i_s,\omega]^{\mathrm{T}}$$

$$\boldsymbol{A}_{11}=\frac{f}{J},\quad \boldsymbol{A}_{12}=\begin{bmatrix}\frac{C}{J} & 0 & -\frac{2C}{J}\end{bmatrix},\quad \boldsymbol{B}_1=[0 \quad 0]$$

$$\boldsymbol{A}_{21}=\begin{bmatrix}-\frac{J}{J_{\mathrm{M}}}\\ 0\\ 1\end{bmatrix},\quad \boldsymbol{A}_{22}=\begin{bmatrix}-\frac{f_{\mathrm{M}}}{J_{\mathrm{M}}} & \frac{C_{\mathrm{m}}}{J_{\mathrm{M}}} & -\frac{f}{J_{\mathrm{M}}}\\ -\frac{C_{\mathrm{e}}}{L} & -\frac{R}{L} & 0\\ 0 & 0 & 0\end{bmatrix},\quad \boldsymbol{B}_2=\begin{bmatrix}0 & \frac{1}{J_{\mathrm{M}}}\\ \frac{1}{L} & 0\\ 0 & 0\end{bmatrix}$$

待定系数矩阵为

$$\boldsymbol{G}_1=[g_{11} \quad g_{12} \quad g_{13}]$$

注意到：$\widetilde{\boldsymbol{B}}_1=\boldsymbol{B}_1-\boldsymbol{G}_1\boldsymbol{B}_2=\begin{bmatrix}0 & \frac{g_{11}}{J_{\mathrm{M}}}\end{bmatrix}$，为避免观测器的输入需要 M_{c} 信号(因为扰动信号 M_{c} 是不能测量的)，因此只能规定 $g_{11}=0$。

将 $\boldsymbol{G}_1$ 代入观测器方程并进行适当整理，从而得到观测器的方程为

$$\begin{cases}\dot{z}=g_1z+g_2\omega_s+g_3\omega\\ \hat{\varepsilon}=z-\left(g_1+\frac{f}{J}\right)\omega\end{cases}$$

其中 $g_1=\boldsymbol{A}_{11}-\boldsymbol{G}_1\boldsymbol{A}_{21}=-\frac{f}{J}-g_{13}$，$g_2=\frac{C}{J}$，$g_3=-\left(g_{13}^2+g_{13}\frac{f}{J}+\frac{2C}{J}\right)$。

值得注意的是，观测器的结构参数仅与天线及传动机构的结构参数有关，而与电机的电气参数如 L，R，C_{e}，C_m 等无关，与控制 u_s 及扰动力矩 M_{c} 也无关。一般来说，天线的转动惯量 J，传动机构的弹性系数 C 不易发生剧烈变化，因而观测器的性能是比较稳定的。

特别是，天线的摩擦系数往往是可以忽略的，这样，观测器方程的形式更为简单

$$\begin{cases}\dot{z}=-g_{13}z+\frac{C}{J}\omega_s-\left(g_{13}^2+\frac{2C}{J}\right)\omega\\ \hat{\varepsilon}=z+g_{13}\omega\end{cases}\tag{4.92}$$

g_{13} 实际上是观测器的极点，由设计人员根据对观测器响应速度的要求确定。

(5) 差速子空间

较之单电机驱动系统，双电机驱动的系统复杂性在于系统中存在一个差速子空间，其状态方程为

$$\begin{bmatrix}\dot{\theta}_{\mathrm{d}}\\ \dot{\omega}_{\mathrm{d}}\\ \dot{i}_{\mathrm{d}}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0\\ -\dfrac{C}{J_{\mathrm{M}}} & -\dfrac{f_{\mathrm{M}}}{J_{\mathrm{M}}} & \dfrac{C_{\mathrm{m}}}{J_{\mathrm{M}}}\\ 0 & -\dfrac{C_{\mathrm{e}}}{L} & -\dfrac{R}{L}\end{bmatrix}\begin{bmatrix}\theta_{\mathrm{d}}\\ \omega_{\mathrm{d}}\\ i_{\mathrm{d}}\end{bmatrix}+\begin{bmatrix}0\\ 0\\ \dfrac{1}{L}\end{bmatrix}(u_1-u_2) \tag{4.93}$$

该子系统虽然与和速子空间是解耦的，对机械谐振不起作用，但是仍然会增加系统能量的损耗和加剧机械磨损。而且从实际系统运行的情况来看，该子空间阻尼很小，振荡频率较低，衰减很满，因此必须增加差速振荡的阻尼。

当 $u_1\neq u_2$ 时，差速子空间通过系数矩阵 $\boldsymbol{B}_{\mathrm{d}}=(0,0,1/L)^{\mathrm{T}}$ 作用于该子空间。不难验证，$\mathrm{rank}(\boldsymbol{B}_{\mathrm{d}}\vdots\boldsymbol{A}_{\mathrm{d}}\boldsymbol{B}_{\mathrm{d}}\vdots\boldsymbol{A}_{\mathrm{d}}^2\boldsymbol{B}_{\mathrm{d}})=3$，其中 $\boldsymbol{A}_{\mathrm{d}}$ 为差速子空间的系数矩阵。即该子系统是完全能控的，理论上可以实现全状态反馈控制。然而，实际系统一般不对 θ_1,θ_2 测量，同时也考虑到不降低系统的静差度，因此只可引入 i_{d} 与 ω_{d} 的反馈。此时虽然不能对差速子空间任意配置极点，但对三阶系统来说，引入两个状态变量的反馈仍可以达到相当程度的阻尼效果，即

$$u_{\mathrm{d}}=u_1-u_2=(0,k_{\mathrm{d}\omega},k_{\mathrm{d}i})(\theta_{\mathrm{d}},\omega_{\mathrm{d}},i_{\mathrm{d}})^{\mathrm{T}}=k_{\mathrm{d}\omega}\omega_{\mathrm{d}}+k_{\mathrm{d}i}i_{\mathrm{d}}$$

从而得到反馈后的差速子系统的系数矩阵为

$$\widetilde{\boldsymbol{A}}_{\mathrm{d}}=\boldsymbol{A}_{\mathrm{d}}-\boldsymbol{B}_{\mathrm{d}}(0,k_{\mathrm{d}\omega},k_{\mathrm{d}i})=\begin{bmatrix}0 & 1 & 0\\ -\dfrac{C}{J_{\mathrm{M}}} & -\dfrac{f_{\mathrm{M}}}{J_{\mathrm{M}}} & -\dfrac{C_{\mathrm{m}}}{J_{\mathrm{M}}}\\ 0 & -\dfrac{C_{\mathrm{e}}+k_{\mathrm{d}\omega}}{L} & -\dfrac{R+k_{\mathrm{d}i}}{L}\end{bmatrix}$$

特征方程为

$$\begin{aligned}\det(s\boldsymbol{I}-\widetilde{\boldsymbol{A}}_{\mathrm{d}})=&s^3+\left(\frac{f_{\mathrm{M}}}{J_{\mathrm{M}}}+\frac{R+k_{\mathrm{d}i}}{L}\right)s^2\\&+\left(\frac{f_{\mathrm{M}}(R+k_{\mathrm{d}i})+C_{\mathrm{m}}(C_{\mathrm{e}}+k_{\mathrm{d}\omega})+CL}{J_{\mathrm{M}}L}\right)s+\frac{(R+k_{\mathrm{d}i})C}{J_{\mathrm{M}}L}=0\end{aligned}$$

适当选择 $k_{\mathrm{d}\omega},k_{\mathrm{d}i}$，以保证系统有足够阻尼，达到有效消除差速振荡的目的。差速子系统的模型设计完成后，$k_{\mathrm{d}\omega},k_{\mathrm{d}i}$ 的值则通过仿真计算来确定。

(6) 随动系统的结构

前面我们讨论了双电机驱动的伺服系统模型设计，说明了全状态反馈控制下和速子空间的模型(包括天线加速度观测器模型)，设计了差速子空间的模型，并说明了这两个子空间在系统参数对称的情况下完全解耦这一特点，因而可以分别设计。

工程上习惯于将雷达随动系统分为两个环路：一是如前所述的速度环，它包括驱动电机到天线及其校正环节；另一是位置环，它包括除接收机以外的有关位置信号的校正环节。为了实现雷达的准确跟踪，除了速度环要克服机械谐振并具有足够的响应速度外，还要求有足够的静态精度，即除了要求对位置信号跟踪无静差外，还要求对速度信号跟踪无静差，因此位置环一般均采用比例积分校正。

这样就可得到带天线加速度观测器的随动系统全状态反馈的大型雷达随动系统的模型(未包括差速子系统)的结构图，如图 4.37 所示。

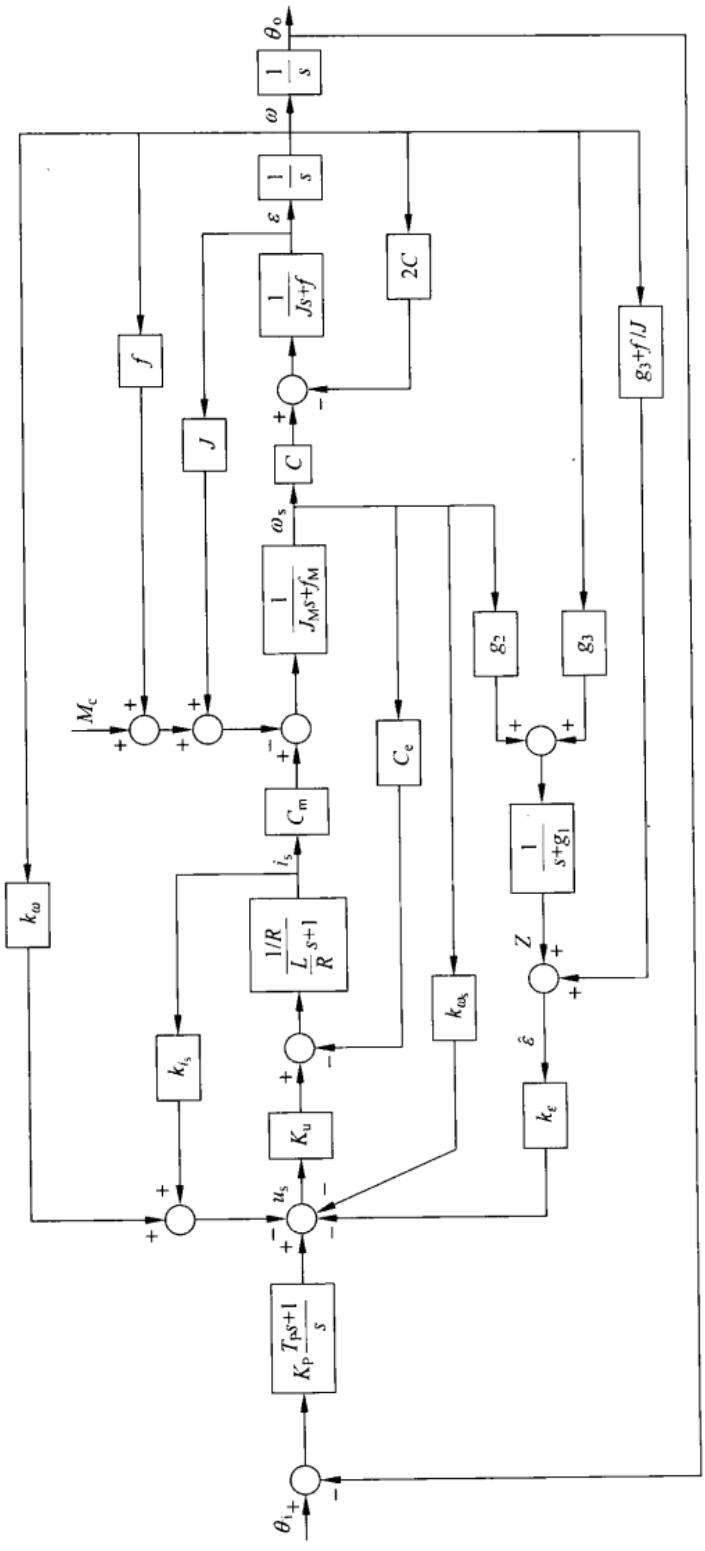


图 4.37 大型雷达随动系统的模型设计

本节对大型雷达控制系统建立了结构有效(Structurally Valid)的模型，从系统的构成、子系统的内部结构、子系统之间的相互作用等均与实际系统相匹配。这样，模型不但能用于观测实际系统的行为，而且能反映实际系统产生行为的操作过程。它反映了系统的本质过程，满足相似性原则。基于简单性原则，该模型忽略了系统中存在的某些非线性环节，它可以在进一步仿真研究中加以添加。这说明，根据仿真的目的进行模型设计是十分重要的。

4.4.3 仿真设计

1. 线性系统仿真设计

从图 4.37 可以看到，大型雷达控制系统是比较复杂的，需要设计的内容包括：全状态反馈控制律的确定、观测器参数的确定、位置环控制器参数的确定等。这些均需要仿真技术的支持加以解决。

本小节首先基于上述线性模型进行设计，下一小节在此基础上增加考虑非线性后再做进一步确认设计。

某实际雷达系统的参数如下：天线直径 10m，机械谐振频率为 7Hz，相对阻尼系数为 0.017 2。按经典的双环(电机速度环、电流环)调节思想设计，随动系统的开环剪切频率为 5.6rad/s，加速度误差系数为 5.6，系统的跟踪能力及抗干扰能力较差，原因是机械谐振没有有效的克服。图 4.38 是采用双环调节思想设计的典型过渡过程曲线，机械谐振十分明显。

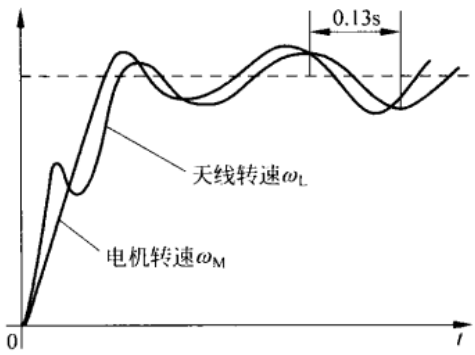


图 4.38 机械谐振引起的过渡过程示意

(1) 机械谐振的消除

消除机械谐振的关键在于和速子空间的设计。和速子空间的状态方程已在式(4.91)中给出。先不考虑扰动 M_c，并认为天线加速度是可以得到的。从而其状态方程如下

$$\begin{bmatrix} \dot{i}_s \\ \dot{\omega}_s \\ \dot{\varepsilon} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{C_e}{L} & 0 & 0 \\ \frac{C_m}{J_M} & -\frac{f_M}{J_M} & -\frac{J}{J_M} & -\frac{f}{J_M} \\ 0 & \frac{C}{J} & -\frac{f}{J} & -\frac{2C}{J} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_s \\ \omega_s \\ \varepsilon \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{K_u}{L} \\ 0 \\ 0 \\ 0 \end{bmatrix} u_s \tag{4.94}$$

式中增加了一个参数 K_u，是考虑到反馈控制信号 u_s 必须进行前置放大以驱动控制回路。

线性系统全状态反馈控制设计的方法很多，这里介绍应用比较广泛的二次型性能指标下的最佳控制设计。

对于系统

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$

若取目标函数为二次型性能指标

$$J = \frac{1}{2}\int_0^{\infty} (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u})\mathrm{d}t$$

其中 $\boldsymbol{Q}$、$\boldsymbol{R}$ 为加权阵，$\boldsymbol{Q}$ 为半正定阵，$\boldsymbol{R}$ 为正定阵。则其最佳控制为系统全状态线性反馈

$$\boldsymbol{u} = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}$$

反馈系数阵为

$$\boldsymbol{K} = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}$$

$\boldsymbol{P}$ 为 Riccate 代数方程的解

$$\boldsymbol{P}\boldsymbol{A} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{P} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P} + \boldsymbol{Q} = 0$$

选择不同的 $\boldsymbol{Q}$、$\boldsymbol{R}$ 可得到最佳控制解，解得的 $\boldsymbol{P}$ 阵为正定对称阵，这样就将反馈系数的计算转化为求解 Riccate 代数方程。求解 Riccate 代数方程的方法很多，这里就不一一介绍了。

表 4.9 是某一实际雷达系统的结构参数。

表 4.9 某一实际雷达系统的结构参数

结构参数	数值	单位	结构参数	数值	单位
R	2	Ω	C_{m}	0.055	kg·m/A
L	0.25	H	J_{M}	5.66×10^{-4}	kg·m/s²
K	100	无量纲	f_{M}	0.001 9	无量纲
C_{e}	0.544	V/s	C	0.517	kg·m/rad
J	11.856×10^{-4}	kg·m/s²	f	0	

通过选择不同的 $\boldsymbol{Q}$、$\boldsymbol{R}$，得到相应的反馈系数，通过仿真计算确定的方案如下：

$$k_{i_s} = 0.217\,136, k_{\omega_s} = 0.114\,916, k_{\varepsilon} = 0.005\,717, k_{\omega} = 0.057\,169$$

表 4.10 给出了该状态方程在全状态反馈下系统特征根与开环系统及典型双环调节的特征根的比较结果，表中对应的系统传递函数为

$$G(s) = \frac{k_u\omega_1^2\omega_2^2}{(s^2 + 2\zeta_1\omega_1 s + \omega_1^2)(s^2 + 2\zeta_2\omega_2 s + \omega_2^2)}$$

表 4.10 和速子空间系统性能比较

	ω_1/rad/s	ζ_1	ω_2/rad/s	ζ_2	过渡过程时间(5%)
开环系统	10.6	0.433	43.5	0.017 2	
双环调节系统	27.1	0.671	49.2	0.214	0.21
全状态反馈	46.2	0.866	59.6	0.27	0.118

从表 4.10 可以看到，原开环系统的机械谐振频率为 43.5rad/s，相对阻尼系数为 0.017 2，谐振严重。按经典控制理论设计的双环调节，虽然加大了机械谐振这一对极点的阻尼，但极点的位置改变不够大(49.2rad/s)，全状态反馈理论上是可以任意配置极点，但 Q 值的选择必须考虑到实际实现(这里不再详细讨论)，从所得到的结果来看，机械谐振这一对极点移到了 59.6rad/s，阻尼系数达到 0.27，使得系统具有较快的响应速度，而接近原机械谐振频率的另一对极点(46.2rad/s)的阻尼系数达到 0.866，有效地克服了机械谐振，从而使其过渡过程时间从双环调节的 0.21s 缩短到 0.118s。

(2) 位置环的仿真设计

在速度环仿真设计完成后，还应对位置环进行设计，因为，速度环克服机械谐振的程度最终要满足天线对位置跟踪的要求。位置环为实现无静差的要求，一般采用比例积分控制 $K_P(T_P s+1)/s$。有两个参数需要通过仿真整定，即积分时间常数 T_P 及放大倍数 K_P。这可以根据经典控制理论事先确定初值，然后通过多次仿真比较加以确定，也可以采用以过渡过程时间、超调量作为性能指标，选择目标函数进行基于优化的仿真，这相对比较简单，这里只给出仿真结果：在前述速度环的参数下，对线性系统来说，比例积分控制器的时间常数 $T_P=0.273$s，放大倍数 $K_P=18.95$，过渡过程时间为 0.59s，超调量为 29%，满足实际系统设计的要求。

(3) 加速度观测器的设计

前面在确定全状态反馈系数时天线加速度 ε 是按能直接测量来考虑的。根据工程应用的要求，采用角速度观测器来实现。观测器的输出总会带来一些延迟，因此必须通过仿真来确定观测器的极点，整定其结构参数，使得系统的性能尽可能接近直接测量 ε 条件下的结果。

观测器的极点 g_1 一般可按原系统各环节中最小时间常数的倒数的 5 倍左右初定。本系统最小时间常数的环节为电枢回路的时间常数 L/R，根据表 4.9 中给出的参数计算得到 $L/R=0.125$，即可初选 $g_1=-40$。考虑到观测器的输入信号是 i_s，$\omega_s=\omega_1+\omega_2$，$\omega$，后两个变量是经过测速机滤波后的信号，因此 g_1 应稍微加大，经仿真确定为 $g_1=-50$，从而结合原系统的结构参数，如果忽略天线轴滑动摩擦系数，则天线加速度观测器结构参数如下

$$g_1=-50,\quad g_2=436.1,\quad g_3=-3\,372.43$$

仿真表明按上述参数设计的加速度观测器，可很好地满足系统的要求，图 4.39 及图 4.40 分别给出了在突加给定与突加扰动(M_c)两种情况下理论 ε 与观测器输出的 $\hat{\varepsilon}$ 仿真结果。从图中可以看到，对阶跃给定信号，$\hat{\varepsilon}$ 略滞后 ε，这取决于 g_1 的值，而幅值则是非常精确地重构了 ε，误差小于 5%。对阶跃扰动信号 M_c，在开始阶段理论 ε 与 $\hat{\varepsilon}$ 相差较大，这是由于观测器的设计是按 $dM_c/dt=0$ 即 M_c 是恒定值考虑引起的。这段时间占整个过渡过程时间的 1/5 左右。尽管如此，用 $\hat{\varepsilon}$ 代替 ε 反馈，对系统的影响仍然不大，表 4.11 给出了两种情况下的仿真结果。这表明，尽管作了 $dM_c/dt=0$ 的假定，当 $dM_c/dt\neq 0$，即在非恒值扰动时，采用观测器是可行的。

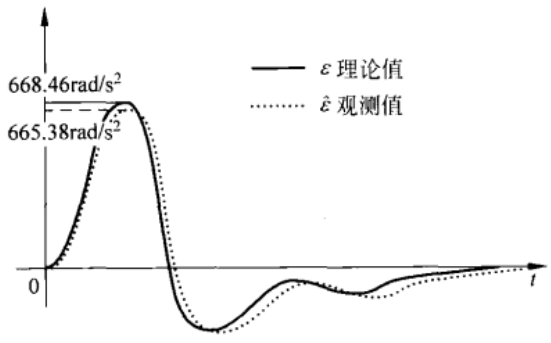


图 4.39 阶跃给定信号下 ε 与 $\hat{\varepsilon}$ 的比较

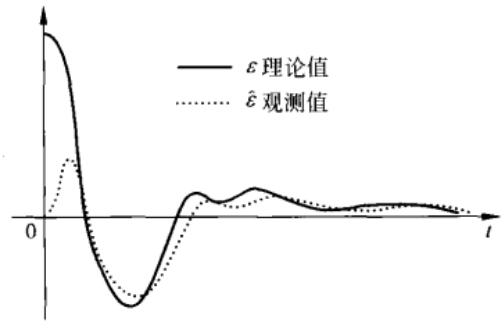


图 4.40 阶跃扰动信号下 ε 与 $\hat{\varepsilon}$ 的比较

表 4.11 采用 ε 与 $\hat{\varepsilon}$ 反馈系统性能比较表

信号类型	信号幅度	采用 ε 反馈		采用 $\hat{\varepsilon}$ 反馈	
		调整时间	超调量	调整时间	超调量
突加给定	小给定信号 $\theta_i=0.424$mil	0.59s	29%	0.6s	27.2%
	大给定信号(*)$\theta_i=4.24$mil	0.59s	58%	0.6s	49.6%
突加扰动	小扰动 $M_c=0.3$kg·m	0.25s	0.16mil	0.28s	0.19mil
	大扰动(*)$M_c=3.0$kg·m	0.40s	1.94mil	0.42s	2.02mil

(*)如果都是线性系统,大给定信号与小给定信号的性能是一样的,这里给出的是考虑非线性环节的情况下(参见后续节)的仿真结果。

2. 考虑非线性及小时间常数的仿真设计

在进行任何实际系统的确认详细设计时,模型设计应该考虑非线性因素的影响以及小时间常数的影响等。

本系统小时间常数环节主要包括晶闸管的延迟时间、测速机滤波时间常数等。非线性主要包括放大器的饱和非线性、传动机构的齿轮间隙非线性、电动机摩擦非线性、晶闸管为防误触发所设定的失灵区、为消除齿隙专门设计的梯形非线性等。

另外,在模型设计时,假设两电机参数是完全对称的,从而得到和速与差速子空间解耦的结论,因而两个子系统可独立分别设计。然而,实际系统并非完全如此,这种不对称采用理论分析十分困难。只有通过仿真才能得到明确的结论,譬如说,考虑两电机参数相差5%左右的情况下,系统方案的可行性还需要进一步论证。

基于以上考虑,需要对原线性系统的模型进行进一步设计。其具体过程不再赘述,只给出相应的结果如下:

1) 系统参数

前面已经给出了部分系统参数及设计参数,那是按和速子空间的设计要求定义的。在考虑非线性、小时间常数及不对称性情况下,模型描述发生变化,为此一并将系统中的参数列在表4.12中。下标1,2分别表示电机1,2及其控制回路的参数。

表 4.12 大型雷达系统仿真设计模型的参数表

结构参数名称	符号	数值	单位
晶闸管放大倍数	K_{s1},K_{s2}	27.9	无量纲
晶闸管时间常数	T_{s1},T_{s2}	0.003 3	s
电机电枢回路电阻	R_1,R_2	2	Ω
电机电枢回路电感	L_1,L_2	0.25	H
电机电势系数	C_{e1},C_{e2}	0.544	V/s
电机电磁力矩系数	C_{m1},C_{m2}	0.055	kg·m/A
电机转动惯量	J_{M1},J_{M2}	5.66×10^{-4}	kg·m/s^2
电机转动摩擦系数	f_{M1},f_{M2}	0.001 9	无量纲
天线转动惯量(折合到电机轴)	J	11.856×10^{-4}	kg·m/s^2
传动机构弹性系数(折合到电机轴)	C	0.517	kg·m/rad
天线转动摩擦系数	f	0	无量纲
测速机时间常数	T_ω	0.003 4	s
信号放大器倍数	K_n	3.584	无量纲
消隙放大倍数	K_T	1.06	无量纲

续表

系统变量	符号	数值	单位
电机电流	i_1，i_2	动态变化	A
电机转速	ω_1，ω_2	动态变化	rad/s
天线加速度(折合到电机轴)	ε	动态变化	rad/s^2
天线转速(折合到电机轴)	ω	动态变化	rad/s
天线转角(折合到电机轴)	θ_o	动态变化	rad
输入角度	θ_i	阶跃	rad
风扰力矩	M_c	阶跃	kg
设计参数	符号	数值	单位
电流反馈系数	k_{i1}，k_{i2}	0.251 22	V/A
电机转速反馈系数	$k_{\omega 1}$，$k_{\omega 2}$	0.150 58	V·s/rad
天线加速度反馈系数	k_ε	0.009 20	V·s^2/rad
天线加速度观测器极点	g_1	−50	1/s
天线转速反馈系数	k_ω	0.114 09	V·s/rad
位置环积分时间常数	T_P	0.273	s
位置环比例系数	K_P	18.95	无量纲

2) 系统中非线性环节及参数

非线性环节说明如图 4.41 所示，其中(a)表示信号及电压放大限幅，(b)表示晶闸管失灵区非线性，(c)表示消隙非线性，(d)表示传动机构齿隙，(e)表示电机摩擦系数非线性(其中标出的 f 为近似线性滑动摩擦系数)。

所对应系统结构图如图 4.42 所示。对于该图所描述的系统，既包括了线性部分，也包括了非线性部分。

3) 非线性系统仿真实验设计

对一个系统进行仿真时，必须根据仿真的目的和模型的特点进行实验设计。在前面讨论线性系统仿真设计时，没有特别地对仿真设计加以说明的原因是，其实验设计相对比较简单。当模型比较复杂，仿真的目的与模型运行有较大关系时，仿真实验设计是必要的。

(1) 仿真计算要求

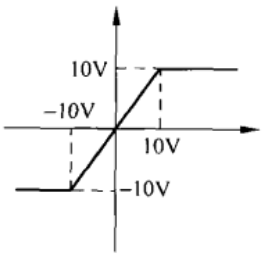
对非线性系统，仿真计算是在考虑非线性、小时间常数及两电机参数不对称的情况下，分别加小给定、小扰动，即部分环节处在线性工作区的情况下，以及分别加大给定、大扰动，即所有非线性环节均起作用的情况下，系统的性能指标是否能达到要求。

(2) 实验要求

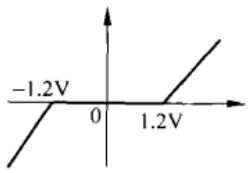
系统的初始条件为 0，此时消隙非线性环节自动作用，必须等消隙非线性作用完成后才能分别加大给定、大扰动。若考虑系统性能指标采用单位阶跃函数作用下系统的超调量及过渡过程时间，那么无论是给定或扰动必须在仿真启动 0.85s 后才能作用到系统上，因此，按延迟 0.85s 的单位阶跃给定输入，输出过渡过程 θ_o 曲线，然后加延迟 0.85s 单位阶跃扰动 M_c，输出过渡过程 θ_o 曲线。观察仿真结果，是否满足要求。给定信号与扰动信号如图 4.43 (为延迟 0.85s 的单位阶跃)。

加给定或扰动，这个过程大约需要 0.85s 的时间才能完成。

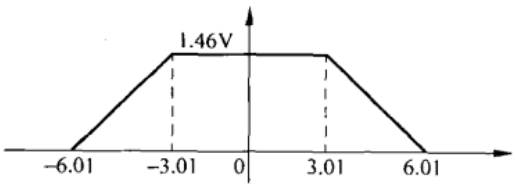
另外，仿真步长的选择是十分重要的，必须保证计算的稳定性，同时既要满足计算精度



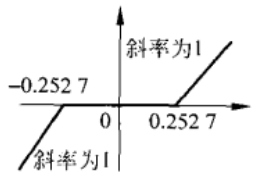
(a) 信号与电压放大限幅



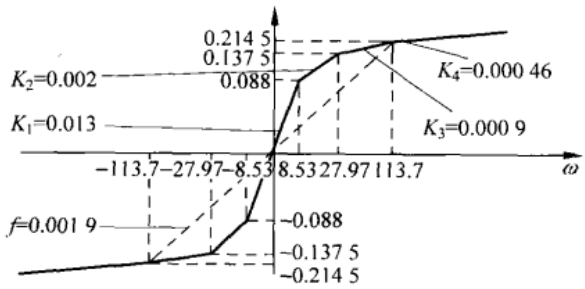
(b) 晶闸管失灵区



(c) 消隙非线性



(d) 传动机构齿隙



(e) 电机摩擦系数非线性

图 4.41 系统中非线性环节及参数

的要求，还要考虑计算速度。经典的数值积分法的稳定域与步长的关系已经在第 3 章中进行了讨论，离散相似法的稳定性在前面几节中也分别作了介绍。本节讨论的是离散相似法的应用，要根据系统中最小时间常数来进行初选。一般初选步长应是最小时间常数的 1/5～1/3。本系统的最小时间常数为 0.003 3，不妨先用 0.000 5～0.001 进行试算。

仿真结果输出要根据研究的目标加以确定。可以将感兴趣的变量信息存储在一个文件中，得到一个仿真过程系统在某一输入作用下内部变量随时间变化的动态信息，以便进行模型分析活动。对本系统，我们关心的是随动系统存在非线性环节的情况下的跟踪性能，包括其在阶跃给定输入下的调整时间、超调量，在阶跃扰动下的调整时间及扰动偏差等，因此，天线的位置 θ_0、天线转速 ω、天线加速度 ε 以及电机的转速 ω_1，ω_2，电流 i_1，i_2 等变量的变化过程是应该输出的。当然，在仿真的不同阶段，感兴趣的变量会有变化，好在仿真系统可以很方便地根据用户的要求来定义系统中的所有变量加以输出。

输出间隔定义也要适当。显然，输出间隔最小值是仿真步长，但如果按步长定义输出间隔，往往需要占用不少机时和内存，一般可考虑计算若干步，譬如 5～10 步输出一次，这样既保证了计算稳定性及计算精度的要求，又能提高仿真的速度。对本系统，输出间隔可考虑为 0.005～0.01。基于上述讨论，可确定如表 4.13 的实验方案。

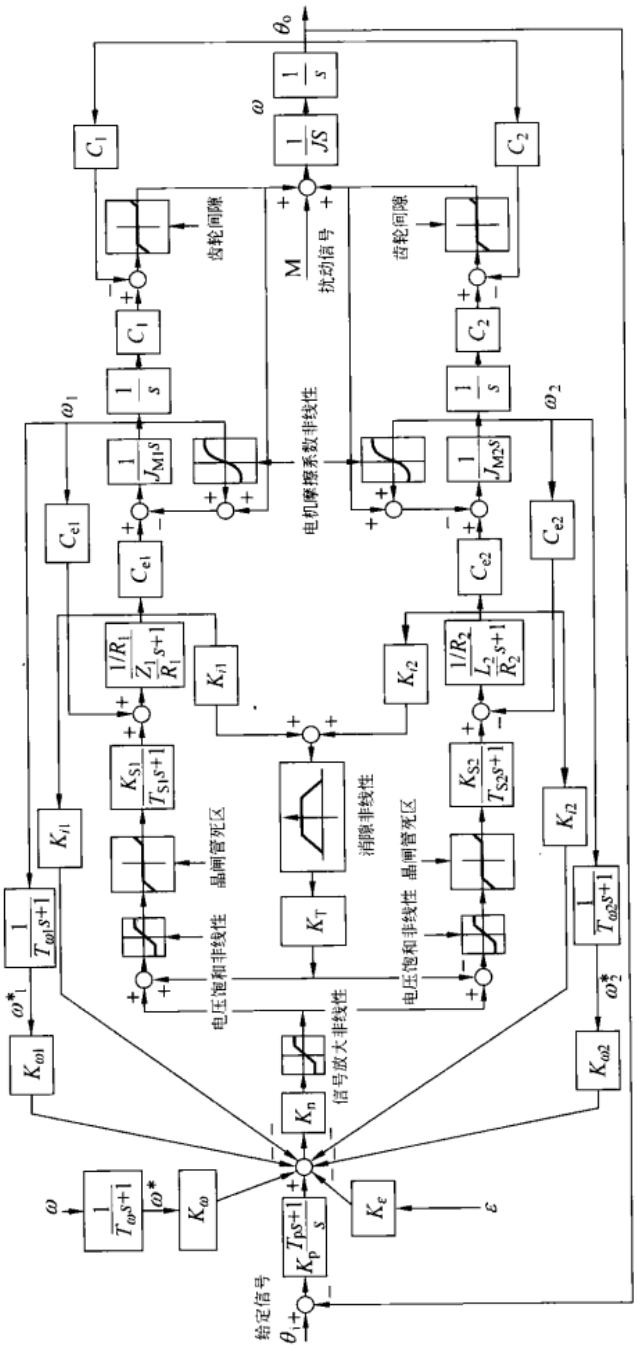
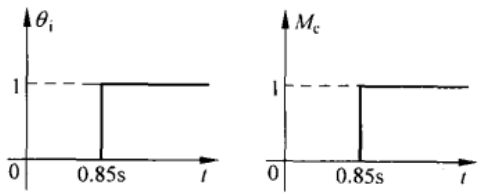


图 4.42 包括非线性的雷达控制系统结构图



(a) 给定信号θ_i (b) 扰动信号M_c

图 4.43 延迟的单位阶跃给定和扰动

表 4.13 大型雷达随动系统仿真设计实验方案

序号	突加给定		突加扰动	
	小给定	大给定	小扰动	大扰动
1	电机参数对称条件下消隙非线性仿真测试，输出变量：电机电流、转速、天线转角			
2	电机参数对称、无差速反馈、天线加速度直接反馈时，系统性能仿真 输出变量：电机电流、转速、天线加速度、天线转速、天线转角			
3	电机参数对称、无差速反馈、天线加速度观测值反馈时，系统性能仿真 输出变量：电机电流、转速、天线加速度(理论值、观测值)、天线转速、天线转角			
4	电机参数相差5%，消隙非线性仿真测试，输出变量：电机电流、转速、天线转角			
5	电机参数相差5%、无差速反馈、天线加速度观测值反馈时，系统性能仿真 输出变量：电机电流、转速、天线加速度(理论值、观测值)、天线转速、天线转角			

(3) 仿真软件选择

对于本节讨论的系统仿真，有多种商用仿真软件可供选择，典型的如 MATLAB。关于如何用 MATLAB 仿真将在第 6 章中进行介绍。这里介绍如何用 4.2.3 节中介绍的非线性离散相似法 CSS02 来进行仿真，下面对其简单地加以说明。

CSS02 的仿真原理及流程已经在 4.2.3 节进行了讨论，它是面向结构图模型的，要求用户首先对结构图进行环节定义，输入计算步距 $STEP$，打印间隔 L_1(即每隔 $L_1\times STEP$ 打印一点)，打印点数 L_2(即仿真总时间为 $STEP\times L_1\times L_2$)，系统的环节数 N，阶跃函数的幅度 R，打印变量的个数(用变量 N_0 表示)，打印变量的编号(最多三个，用变量 N_2，N_3，N_4 表示)。

正如 4.2.3 节中介绍的，每个典型环节包括一个线性块$[(C+Ds)/(A+Bs)]$，其前可附属一个非线性块，其后可附属一个非线性块，为此，CSS02 需要定义 8 个参数：A，B，C，D，X，Y，Z，S。其中，A，B，C，D 为环节的参数，X，Y 是状态变量及输出变量的初值，Z 表示附属非线性环节类别，$Z=0$ 表示该环节无附属非线性。Z 为奇数表示其前附属有非线性，Z 为偶数表示其后附属有非线性。CSS02 中提供了几类典型的非线性，例如 $Z=1$ 表示该环节前有饱和非线性，$Z=2$ 表示该环节后有饱和非线性；$Z=3$ 表示该环节前有回环非线性，$Z=4$ 表示该环节后有回环非线性；$Z=5$ 表示该环节前有失灵区非线性，$Z=6$ 表示该环节后有失灵区非线性。S 定义附属非线性环节的特征参数，每个非线性环节只有一个特征参数，$S=0$ 表示无特征参数，关于 S 值的定义请参见 4.2.3 节。本模型中还有一些非线性环节是原 CSS02 程序中未包括的，如消隙非线性、电机摩擦系数非线性，需要添加相应程序模块。

需要说明的是，CSS02 只容许一个线性块前后至多附属一个非线性块，如果有两个以上非线性块紧邻，又难以分别附属，这时可在两个非线性之间增加一个比例(值为 1)线性块。

(4) 仿真执行

完成了模型设计及实验设计后，就可以执行仿真了，表 4.14 给出 CSS02 某一次运行的输出结果。

表 4.14 大型雷达随动系统仿真运行输出(计算步距 $STEP=0.005$，打印间隔 $L_1 \times STEP=0.01$)

T	电机电流 i	电机转速 ω	天线转角 θ_0
0	0	0	0
0.01	0.164 947	0.017 753 8	0
0.02	0.425 642	0.239 894	0
0.03	0.662 630	0.644 527	0
0.04	0.871 692	1.169 12	0
0.05	1.052 89	1.762 61	0
0.06	1.207 37	2.385 31	0
0.07	1.336 90	3.007 24	0
0.08	1.443 67	3.606 59	0
0.09	1.530 06	4.168 27	0
0.10	1.598 56	4.682 61	0
0.11	1.651 58	5.144 25	0
0.12	1.691 46	5.551 13	0
0.13	1.720 33	5.903 70	0
0.14	1.740 18	6.204 21	0
0.15	1.752 73	6.456 15	0
0.16	1.759 53	6.660 71	0
0.17	1.765 16	6.417 47	0
0.18	1.781 11	5.671 66	0
0.19	1.815 41	4.582 54	0
0.20	1.872 05	3.318 39	0
0.21	1.951 27	2.037 19	0
0.22	2.050 21	0.871 520	0
0.23	2.163 88	−0.081 178 3	−9.303 8E-10
0.24	2.285 84	−0.762 855	−4.450 74E-09
0.25	2.409 66	−1.154 17	−1.048 56E-08
0.26	2.529 16	−1.269 79	−1.677 2E-08
0.27	2.639 29	−1.150 89	−2.079 52E-08
0.28	2.736 48	−0.856 142	−2.230 4E-08
0.29	2.818 73	−0.452 381	−2.219 83E-08
0.30	2.885 63	−6.086 05E-03	−1.777 78E-08
0.31	2.938 12	0.423 643	−1.174 29E-08
0.32	2.978 17	0.790 085	−3.419 79E-09
0.33	3.008 43	1.061 69	5.129 65E-09
0.34	3.031 84	1.222 79	1.367 91E-08
0.35	3.051 28	1.272 45	2.222 85E-08
0.36	3.069 27	1.221 98	3.077 8E-08
0.37	3.087 84	1.091 58	3.827 13E-08
0.38	3.108 34	0.906 609	4.078 58E-08

续表

T	电机电流 i	电机转速 ω	天线转角 θ_0
0.39	3.131 48	0.693 955	3.827 13E-08
0.40	3.157 36	0.478 900	3.072 77E-08
0.41	3.185 59	0.282 689	1.838 13E-08
0.42	3.215 41	0.120 966	9.404 38E-09
0.43	3.245 87	0.003 073 2	5.381 12E-09
0.44	3.275 96	−0.067 847 5	6.386 94E-09
0.45	3.304 73	−0.094 106 7	1.149 15E-08
0.46	3.331 43	−0.082 274 6	1.880 88E-08
0.47	3.355 49	−0.041 739	2.635 24E-08
0.48	3.376 64	0.167 818	3.163 29E-08
0.49	3.394 85	0.082 617 3	2.977 22E-08
0.50	3.410 28	0.146 264	1.770 24E-08
0.51	3.423 39	0.200 185	3.771 97E-10
0.52	3.434 32	0.239 286	−2.778 57E-08
0.53	3.443 86	0.261 003	−5.700 46E-08
0.54	3.452 40	0.265 136	−7.865 48E-08
0.55	3.460 37	0.253 460	−9.044 8E-08
0.56	3.468 12	0.229 179	−9.668 41E-08
0.57	3.475 88	0.196 324	−9.374 21E-08
0.58	3.483 78	0.159 185	−8.066 65E-08
0.59	3.491 86	0.121 803	−5.951 92E-08
0.60	3.500 07	0.087 576 4	−3.754 21E-08
0.61	3.508 32	0.059 014 1	−1.792 87E-08
0.62	3.516 45	0.037 615 9	−2.690 6E-09
0.63	3.524 32	0.023 884 0	5.607 39E-9
0.64	3.531 80	0.017 448 0	1.113 94E-08
0.65	3.538 77	0.017 259 4	1.619 36E-08
0.66	3.545 15	0.021 811 6	2.122 27E-08
0.67	3.550 91	0.029 380 6	2.625 18E-08
0.68	3.556 06	0.382 519 0	3.150 72E-08
0.69	3.562 62	0.046 896 6	3.719E-08
0.70	3.564 66	0.540 960	3.874 9E-08
0.71	3.568 236	0.059 021 8	3.271 41E-08
0.72	3.571 52	0.061 253 2	1.966 37E-08
0.73	3.574 50	0.060 747 4	6.688 66E-09
0.74	3.577 29	0.057 781 1	−1.357 87E-09
0.75	3.579 96	0.052 859 9	−4.385 31E-09
0.76	3.582 53	0.046 621 2	−5.883 375E-09
0.77	3.585 04	0.039 745 3	−7.342 47E-09
0.78	3.587 52	0.032 879 5	−6.889 85E-09
0.79	3.589 93	0.026 570 4	−1.357 86E-09
0.80	3.592 28	0.021 214 5	8.272 83E-09

续表

T	电机电流 i	电机转速 ω	天线转角 θ_o
0.81	3.594 56	0.017 047 3	2.034 26E-08
0.82	3.596 73	0.014 152 3	3.492 7E-08
0.83	3.598 78	0.012 471 6	5.096 97E-08
0.84	3.600 70	0.011 832 1	6.454 82E-08
0.85	3.602 47	0.011 989 7	7.561 22E-08
	开始阶跃给定		
0.86	5.660 18	0.044 159	0
0.87	14.283 7	4.206 20	2.232 01E-05
0.88	16.464 3	14.969 0	6.213 81E-05
0.89	6.314 56	21.890 1	3.420 47E-03
0.90	−0.766 469	17.254 9	0.020 933 8
0.91	1.071 10	6.813 76	0.068 599 1
0.92	8.723 26	0.391 268	0.155 871
0.93	13.216 9	2.456 67	0.282 698
0.94	12.460 1	7.221 60	0.438 436
0.95	10.925 2	11.164 7	0.608 15
0.96	9.761 75	14.355	0.777 673
0.97	8.681 72	16.735	0.934 787
0.98	7.772 11	18.054 1	1.069 58
0.99	6.813 71	18.119 6	1.175 09
1.00	5.924 21	16.613 5	1.248 15
1.01	5.151 43	13.450 1	1.289 74
1.02	4.616 10	8.732 26	1.304 45
1.03	4.994 09	3.061 24	1.299 19
1.04	5.763 49	−1.575 64	1.281 55
1.05	6.427 56	−4.738 09	1.258 63
1.06	6.582 89	−6.697 38	1.236 47
1.07	6.431 72	−7.935 61	1.219 82
1.08	6.611 12	−8.517 70	1.211 86
1.09	7.187 88	−8.046 73	1.213 83
1.10	7.604 92	−6.427 79	1.224 75
1.11	7.401 97	−4.216 34	1.241 93
1.12	6.909 41	−2.183 07	1.261 84
1.13	6.442 78	−0.587 079	1.280 96
1.14	6.198 39	0.590 566	1.296 28
1.15	5.912 47	1.460 05	1.305 45
1.16	5.464 51	1.822 73	1.307 03
1.17	5.110 53	1.586 16	1.300 70
1.18	4.953 11	0.942 719	1.287 25
1.19	4.845 05	0.097 388 4	1.268 28
1.20	4.733 11	−0.878 127	1.245 88
1.21	4.700 80	−1.891 93	1.222 35
1.22	4.753 54	−2.777 55	1.199 90
1.23	4.808 73	−3.417 46	1.180 28

续表

T	电机电流 i	电机转速 ω	天线转角 θ_a
		开始阶跃给定	
1.24	4.835 67	−3.787 53	1.164 69
1.25	4.858 03	−3.886 84	1.153 60
1.26	4.869 62	−3.715 58	1.146 87
1.27	4.839 97	−3.314 60	1.143 76
1.28	4.767 86	−2.765 50	1.143 17
1.29	4.674 32	−2.150 36	1.143 77
1.30	4.568 06	−1.537 22	1.144 28
1.31	4.449 60	−0.989 842	1.143 59
1.32	4.330 13	−0.562 597	1.140 92
1.33	4.225 19	−0.284 876	1.135 87
1.34	4.140 99	−0.160 027	1.128 45
1.35	4.077 54	−0.174 768	1.119 03
1.36	4.036 13	−0.303 036	1.108 25
1.37	4.016 86	−0.506 845	1.096 88
1.38	4.014 66	−0.742 479	1.085 72
1.39	4.022 22	−0.970 481	1.075 51
1.40	4.033 65	−1.157 28	1.066 78
1.41	4.044 05	−1.279 48	1.059 84
1.42	4.048 47	−1.324 52	1.054 76
1.43	4.043 42	−1.202 00	1.051 38
1.44	4.028 00	−1.191 93	1.049 35
1.45	4.003 16	−1.041 43	1.048 23
1.46	3.970 79	−0.861 805	1.047 51
1.47	3.933 71	−0.675 548	1.046 73
1.48	3.895 43	−0.503 202	1.045 52
1.49	3.859 34	−0.360 753	1.043 63
1.50	3.828 10	−0.258 256	1.040 98
1.51	3.803 51	−0.199 419	1.037 60
1.52	3.786 38	−0.181 953	1.033 67
1.53	3.776 53	−0.198 604	1.029 42
1.54	3.772 84	−0.238 792	1.025 14
1.55	3.773 56	−0.290 499	1.021 11
1.56	3.776 55	−0.342 024	1.017 56
1.57	3.780 08	−0.383 466	1.014 64
1.58	3.782 09	−0.407 795	1.012 44
1.59	3.781 38	−0.411 411	1.010 94
1.60	3.777 28	−0.394 155	1.010 04

4.4.4 仿真结果分析

根据仿真实验方案进行仿真并得到每次运行的输出。仿真分析是对仿真结果的可信性分析，既包括对模型的校验(verification)，也包括对模型的确认(validation)。

一个模型的可信性可分为三种水平：

(1) 在行为水平上的可信性

即模型是否重现真实系统的行为，例如消隙非线性的目的是要消除传动机构在不同方向转动时的齿隙，其工作机理是：在系统初始状态为零时，就能触发晶闸管给电机以驱动力矩，使两电机按相反方向转动，驱动传动机构也按相反方向转动，直到两电机的力矩平衡，从而达到消隙的目的，而天线应该是保持不动的。仿真分析就要观察电机的电流、转速以及天线转速及转角。

从表 4.14 可以看到，仿真模型是可信的，因为其计算误差在 10^{-8} 量级。电机的消隙稳态电流为 3.6A，与理论计算及实际测量一致。

(2) 在状态结构水平上的可信性

即模型能否与真实系统在状态结构上互相对应。通过这样的模型可以对系统未来行为进行有效的推测。对于用结构图表示的系统，模型一般具有状态结构水平上的对应性，在进行仿真分析时，就要考察模型的状态变量变化过程。

本例所仿真的系统是待实现的，因此无法进行全系统比较，但仍然可以从两个方面进行分析。一方面，克服机械谐振是本系统的主要目标，可以通过仿真结果来分析由机械谐振产生的影响是否得到有效的消除；另一方面，考察加进非线性及小时间常数以及两电机参数不对称后整个设计方案的有效性，如果能达到设计目标，说明系统设计方案是有效的。

(3) 在分解结构水平上的可信性

即模型能否表示出真实系统的内部工作情况。这是模型可信性的最高要求。

本系统的模型已经足够精细地表示未来系统的实现，根据仿真实验方案进行仿真运行，将用于指导未来实际系统实现。例如，通过观察各放大器的输出考察其饱和程度、晶闸管失灵区设置防止误触发的有效性，消隙非线性的控制点确定等。

虽然表 4.13 只给出了部分数据，但从这一部分数据不难看出，消隙非线性是有效的，系统在所设计的参数下，有效地克服了机械谐振，在小给定下，其过渡过程时间在 0.6 秒(按 5%计算)，超调量为 30%，达到了系统设计的要求。下面分别给出典型实验的仿真结果的曲线图，其中图 4.44 是小给定信号下的过渡过程，图 4.45 是小扰动信号下的过渡过程；图 4.46 是大给定信号下的过渡过程，而图 4.47 是大扰动信号下的过渡过程。

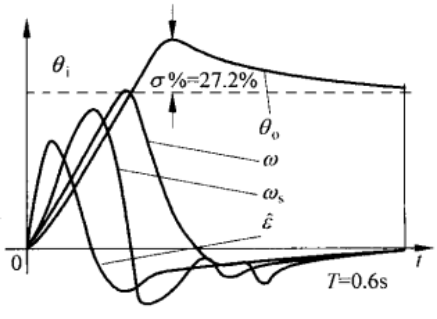


图 4.44 小给定信号下的过渡过程

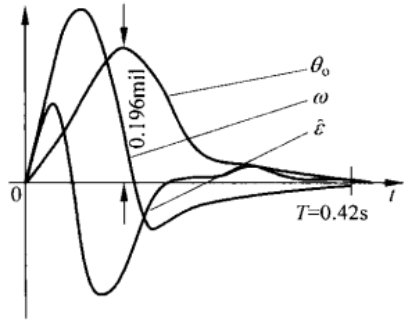


图 4.45 小扰动信号下的过渡过程

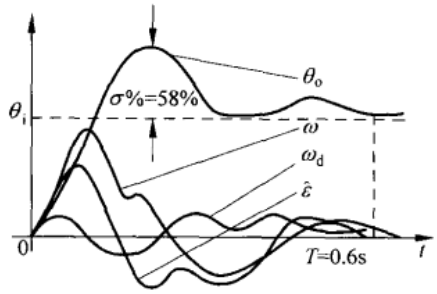


图 4.46 大给定信号下的过渡过程

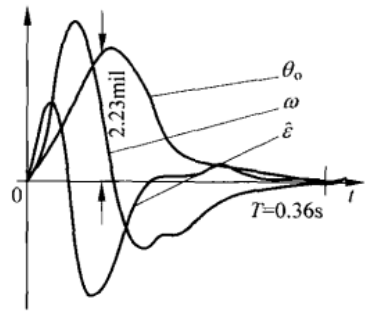


图 4.47 大扰动信号下的过渡过程

在原来线性系统仿真的基础上，系统模型增加考虑了各种非线性、小时间常数及电机系统参数不对称等实际因素，从图中可以看到以下结果：

(1) 小信号输入：在小给定信号($\theta_i=0.424$mil)下，系统控制信号处于线性工作状态，输出 θ_o 的超调量为 27.2%，调整时间为 0.6s，满足设计指标要求，机械谐振已经得到有效克服；在小扰动输入下($M_c=0.3$kg·m)，天线误差只有 0.196mil，调整时间为 0.42s，满足在小扰动下的跟踪精度要求。

(2) 大信号输入：在大给定信号($\theta_i=4.24$mil)下，系统控制信号处于非线性工作状态，输出 θ_o 的超调量为 58%，调整时间仍为 0.6s，满足设计指标要求，机械谐振已经得到有效克服；在大扰动输入下($M_c=3$kg·m)，天线误差只有 2.23mil(<3mil)，调整时间为 0.36s，满足在大扰动下不脱靶的跟踪要求。

值得一提的是，细心的读者可能注意到，在本节的仿真设计过程中，没有对差速子空间再做进一步仿真以确定其控制参数，原因在于，在增加消隙非线性后，差速子空间的阻尼大大增加了，图 4.46 中特别给出了 ω_d 的曲线，其过渡过程与其他的状态变量一样快速地完成了调整，因此就没有必要再增加相关的反馈通路了，从而简化系统的结构。

4.4.5 小结

本节以雷达随动系统设计为例介绍了离散相似法的应用。从本质上讲，CSS02 实现一个系统仿真的思想是，将该系统用虚拟采样开关隔离为若干个子系统，这样整个系统的仿真转化为若干个小系统的仿真，它们在采样步交换信息，实现同步。离散相似法其原理简单，实现方便，精度适当，速度较快，特别适合于控制系统的仿真。用户可以根据需要自由地定义环节，添加虚拟采样开关。然而，正如在 4.1.1 节讨论中看到的那样，任何一个可实现的重构器均会引入相位延迟和幅频误差，因此在实际应用中，虚拟采样/重构器不宜添加太多，步长不宜太大。作为经验值，用离散相似法仿真时，为达到四阶龙格-库塔法同样精度，前者的步长应是后者的 1/10～1/5。

系统设计是一项复杂的任务，从任务的提出，设计思想的确定，初步设计，仿真设计直到样机制造、定型生产，在整个系统开发周期中，仿真无疑是最强有力的工具之一。它可以使理论设计思想加以具体实现与完善，论证设计方案的可行性，为系统实施打下坚固的基础，

它还可以进行各种方案优劣比较，优化设计，而且它还能大大缩短设计周期，具有节省投资、安全可靠等优点，因此仿真技术越来越广泛地应用于新产品、新系统的研制与开发中。

4.5 采样控制系统仿真

采样控制系统（或称数字控制系统）是这样一类控制系统：它首先要对受控对象（过程）的有关信号进行采样，通过输入通道（A/D）把模拟信号转换成数字量，然后将它送给数字控制器或计算机；数字控制器或计算机根据输入的数字信息，按预定的控制规律进行计算；最后将计算结果通过输出通道（D/A）转换成模拟量以控制被控对象，使被控量达到预期指标要求。这类系统既具有连续系统的特征（如受控对象），也具有离散系统的特征（如数字控制器）。然而，从数字仿真的建模方法学角度来看，虽然这类系统有它自身的特点，但它与连续系统没有本质的区别，因而一般将其归类为连续系统仿真。

4.5.1 采样控制系统

图 4.48 表示典型采样控制系统的结构。一个典型的采样控制系统由以下几部分组成：

（1）连续的被控对象或被控过程；

（2）离散的数字控制器；

（3）采样开关或模数转换器；

（4）数模转换器或信号重构器。

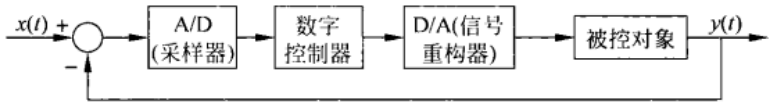


图 4.48 采样控制系统框图

将图 4.48 所示的采样控制系统与图 4.1 所示的离散相似法所得到的系统进行比较，则不难看到，两者的结构是相近的，其被控对象均是连续的，系统中均有采样器和保持器，因此离散相似法可以很方便地用于采样控制系统的仿真。

那么，采样控制系统仿真具有哪些特点呢？

首先，在对连续系统进行离散化时，其采样开关是虚拟的，即其采样间隔、采样器所处位置及保持器的类型是用户根据仿真精度和仿真速度的要求加以确定的。一般来说，在连续系统仿真时，仿真所用的离散化模型中的虚拟采样间隔（即仿真步距）对整个系统来说是唯一的，且是同步的。而采样控制系统则不同，其采样周期、采样器所处位置及保持器的类型是实际存在的。因此，在对采样控制系统进行仿真时，连续部分离散化模型中的仿真步距与实际采样周期可能相同，也可能不同。对于给定的采样控制系统，首先必须解决的是：如何来确定仿真步距？

众所周知，对一个连续模型来讲，不同的仿真步距得到的差分模型不同，其仿真精度也不同。因此仿真步距的选择与仿真方法的选择是紧密相连的。或者说，为实现一定精度与

一定速度的仿真计算，仿真步距与方法的选择必须兼顾考虑。由于实际系统分为离散和连续两部分，从而得到的差分模型也分为两部分，如何处理这两部分模型之间的联系是采样控制系统仿真中的难点。因此，第二个必须解决的问题是：如何处理不同采样间隔下的差分模型。

4.5.2 采样周期与仿真步距

我们规定 T_s 为采样周期，T 为仿真步距。对于图 4.48 所示的典型采样控制系统，可用图 4.49 所示的结构图来表示。其中 $G(s)$ 为被控对象的传递函数，$H(s)$ 为信号重构器的传递函数，$D(z)$ 为数字控制器的 z 传递函数，T_s 是采样控制系统中实际的采样周期，$X(s)$ 为输入信号，$Y(s)$ 为输出信号。

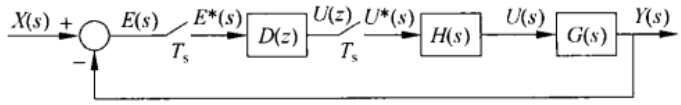


图 4.49 采样控制系统方块图

对图 4.49 所示的采样系统进行仿真，仿真步距的选择必须根据被控对象的结构、采样周期的大小、信号保持器的类型以及仿真精度和仿真速度的要求来综合考虑。一般来说，往往有三种情况：

(1) 采样周期 T_s 与仿真步距 T 相等；

(2) 仿真步距 T 小于采样周期 T_s；

(3) 改变数字控制器的采样间隔 T_s。

下面就每一种情况分别进行讨论。

1. 采样周期 T_s 与仿真步距 T 相等

如果选择仿真步距与采样周期相同，那么在对系统进行仿真时，实际采样开关与虚拟的采样开关在整个系统中均是同步工作的。因此这种仿真与连续系统仿真完全相同，从而可大大简化仿真模型，缩短仿真程序，提高仿真速度。

在什么情况下可考虑仿真步距 T 与采样周期 T_s 相等呢？

如果实际系统中的采样周期 T_s 比较小，系统的阶次比较低，取 $T=T_s$ 可满足仿真精度的要求时，当然应该尽可能选择两者相等。

在对系统中连续部分进行离散化时，虚拟采样开关及信号重构器的数目应尽量少，因为虚拟采样开关及信号重构器对信号幅度和相位引起畸变和延迟，从而带来误差。因此，在连续部分入口加采样器和信号重构器，即将实际系统中的采样器和信号重构器与虚拟的采样信号重构器统一起来，而连续部分 $H(s)G(s)$ 内部不再增加虚拟采样开关和信号重构器。这样在建立连续部分的差分模型时，应计算 $Z\{H(s)G(s)\}=G(z)$，得到仿真模型如图 4.50 所示。

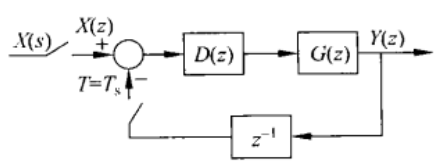


图 4.50 $T=T_s$ 时的仿真模型

众所周知，当系统的阶次较高时，$Z\{H(s)G(s)\}$ 往

往是不易求取的，因此另一种方法是将被控连续对象变成状态空间表达式的形式，即

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}u(t)$$

其中 $u(t)$ 为信号重构器 $H(s)$ 的输出信号。其离散的状态空间表达式为

$$\boldsymbol{x}[(k+1)T_s] = \mathrm{e}^{\boldsymbol{A}T_s}\boldsymbol{x}(kT) + \int_{kT_s}^{(k+1)T_s} \mathrm{e}^{\boldsymbol{A}[(k+1)T_s-\tau]}\boldsymbol{B}u(k\tau)\mathrm{d}\tau$$

特别是当 $H(s)$ 为零阶信号重构器时，可得

$$\begin{aligned}\boldsymbol{x}[(k+1)T_s] &= \mathrm{e}^{\boldsymbol{A}T_s}\boldsymbol{x}(kT) + \int_{kT_s}^{(k+1)T_s} \mathrm{e}^{\boldsymbol{A}[(k+1)T_s-\tau]}\boldsymbol{B}\mathrm{d}\tau \cdot u(kT_s)\\ &= \boldsymbol{\Phi}(T_s)\boldsymbol{x}(kT_s) + \boldsymbol{\Phi}_{\mathrm{m}}(T_s)u(kT_s)\end{aligned}$$

在某些采样控制系统中，T_s 比较大，选择 $T=T_s$ 可能会引起较大的误差。此时，为保证仿真精度，在对 $H(s)G(s)$ 离散化时，必须采取补偿措施以减少因仿真步距过大而引起的误差。

2. 仿真步距 T 小于采样间隔 T_s

这是采样控制系统仿真中最常见的情况。一般说来，采样间隔 T_s 是根据系统频带宽度、实际采样开关硬件的性能和实现数字控制器计算程序的执行时间长短来确定的。由于种种原因（如控制算法比较复杂，数字控制器完成所要求的控制算法需要较长的时间等），采样间隔 T_s 比较大，但系统中连续部分若按采样间隔选择仿真步距 T，将出现较大的误差，因此有必要使 $T<T_s$。

另外，当系统中连续部分存在非线性时，正如离散相似法所考虑的那样，为了便于仿真程序处理，需要将系统分成若干部分分别建立差分模型。此时，就要在各部分的入口设置虚拟采样器及保持器，而每增设一对虚拟采样器和保持器都将引入幅值和相位的误差。为了保证仿真计算有足够的精度，必须缩小仿真步距 T，因此也有必要使 $T<T_s$。

此时，系统仿真模型中将会有两种频率的采样开关：离散部分的采样周期 T_s，连续部分的仿真步距 T。为了便于仿真程序的实现，一般取 $T_s=NT$，其中 N 为正整数。

对这一类仿真系统，要分两部分分别进行仿真计算，对离散部分用采样周期 T_s 进行仿真，对连续部分用仿真步距 T 进行仿真。离散部分每计算一次差分模型，将其输出按保持器的要求保持，然后对连续部分的仿真模型计算 N 次，将第 N 次计算的结果作为连续部分该采样周期的输出。此时，仿真程序的结构如图 4.51 所示。

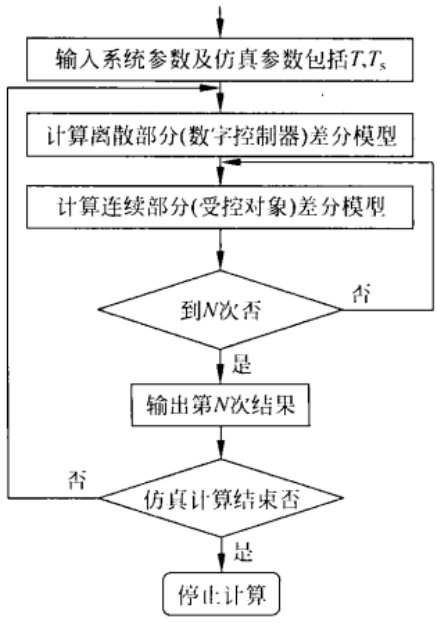


图 4.51 采样控制系统程序框图

系统中存在不同频率的采样开关的另一种情况是：采样系统中有多个回路，且每个回路的采样周期不同。一般内回路的采样周期比较小，而外回路的采样周期比较大。

例如数字控制的双环调速系统，其内环（电流环）反应速度较快，电流控制器的采样频率比较高，而外环（速度环）变化比较缓慢，因而速度调节器的采样频率比较低。同时这个回路校正运算比较复杂，运算时间较长，也要求采样周期长些。

又如“智能汽车”的自动驾驶系统有内外两个反馈回路，一是驾驶盘执行转角的局部反馈，另一个是汽车位置的偏差反馈。前者由于执行机构固有频率高，显然要求采样周期较短；而后者允许采样周期较长，因为整个汽车动态特性频带较窄。这一类系统的典型结构如图4.52。图中 $D_1(z_1)$，$D_2(z_2)$ 分别是外环和内环数字控制器的 z 传递函数。由于是分别对采样周期 T_{1s} 和 T_{2s} 进行 z 变换，故取不同的算子 z_1，z_2 以示区别。$H_1(s)$ 和 $H_2(s)$ 分别为外环和内环的保持器。

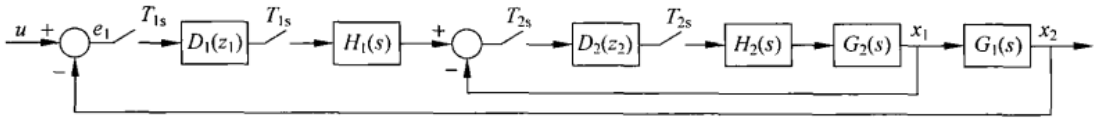


图4.52 双回路采样控制系统

若 $G_1(s)$，$G_2(s)$ 为线性定常系统，并且设 $T_{1s}=nT_{2s}$（其中 n 为正整数），那么，对这类系统进行仿真时，可按内外环路分别取仿真步距进行仿真。此时，若 $G_1(s)$，$G_2(s)$ 不太复杂，可取仿真步距与采样周期相等，即 $T_1=T_{1s}$，$T_2=T_{2s}$。

如果 $G_1(s)$，$G_2(s)$ 比较复杂，或者系统中存在非线性环节，无法对 $H_2(s)G_2(s)$ 和 $G_1(s)$ 进行 z 变换，那么就需要在 $G_1(s)$、$G_2(s)$ 中加进若干个虚拟采样开关和保持器。为保证仿真精度，仿真步距应小于采样周期，即 T_1 应小于 T_{1s}，T_2 应小于 T_{2s}，且令 $T_1=\frac{1}{N}T_{1s}$，$T_2=\frac{1}{M}T_{2s}$，其中 N，M 为二个常数。

系统仿真的过程应该是：首先对 $D_1(z_1)$ 按采样周期 T_{1s} 进行仿真，将其输出经 $H_1(s)$，保持 T_{1s} 时间，然后进行内环仿真。

在进行内环仿真时，首先对 $D_2(z_2)$ 按采样周期 T_{2s} 进行仿真计算，其输出经 $H_2(s)$ 保持 T_{1s} 时间；在 T_{2s} 时间间隔内对 $G_2(s)$ 按仿真步距 T_2 进行计算，计算 M 次后，将第 M 次计算结果作为经 T_{2s} 采样周期后内环的输出。

此时的输出一方面经反馈回路与 $H_1(s)$ 的输出进行比较后送给 $D_2(z_2)$ 以进行下一个 T_{2s} 的 $D_2(z_2)$ 计算；同时，此输出加到 $G_1(s)$ 上。为便于仿真程序处理，可选择 $N=n$，即外环仿真步距 T_1 与内环的采样间隔 T_{2s} 相等，因而还需要对 $G_1(s)$ 按 $T_1=T_{1s}$ 进行仿真计算。这种处理方法实质上是将内环看成是 $G_1(s)$ 的一部分，对该部分按 T_2 进行仿真，也就是系统中有不同的仿真步距。

当内环及 $G_1(s)$ 进行 N 次仿真计算后将第 N 次仿真计算结果进行输出，并经反馈回路与输入信号 $u(s)$ 进行比较后送入 $D_1(z_1)$，再进行下一个采样间隔 T_{1s} 的仿真计算。其仿真

程序的框图如图 4.53。

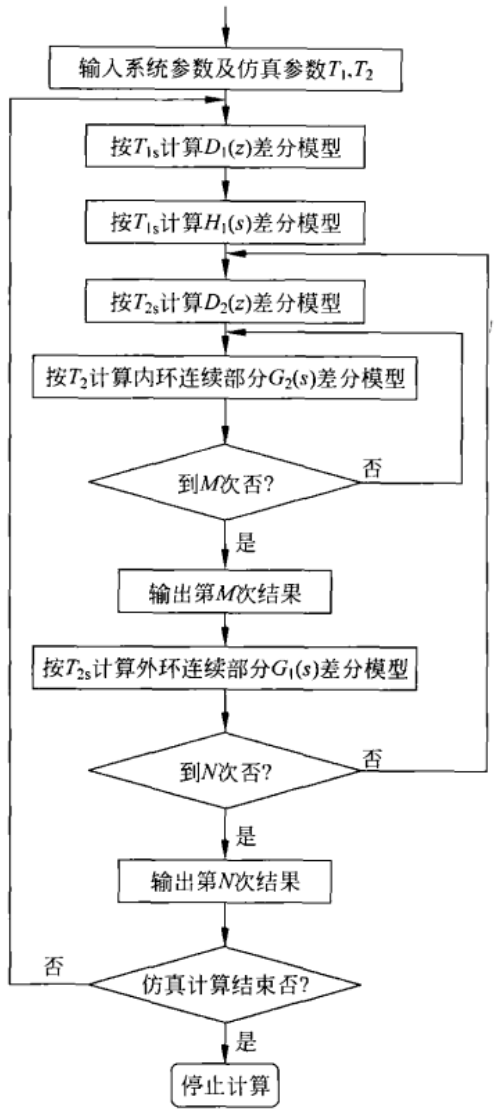


图 4.53 双回路采样控制系统仿真程序框图
($T_1 = T_{2s}$)

3. 不同采样周期差分模型的转换

在对采样控制系统进行仿真时,有时需要对原有的数字控制器的差分模型进行仿真设计,包括修改采样间隔的大小等,那么如何确定在新的采样间隔下数字控制器的差分模型呢?

确定差分模型的原则是:两个脉冲传递函数映射到 s 平面上具有相同的零极点,并且有相同的稳态值,则两个系统是等价的。

根据这一原则,设原采样系统传递函数 $D(z)$,其采样间隔为 T_s,首先将 $D(z)$ 映射到 s 平面上,求得 $D(z)$ 在 s 平面上相应的零极点。然后按新的采样间隔 T_s' 再映射到 z 平面上,求得新的 z 传递函数 $D(z)$。最后,根据稳态增益相等这一原则确定 $D(z)'$ 的增益因子。这也就是根匹配法。

例 4.3 一数字控制器的 z 传递函数为

$$D(z) = 2.62\frac{z-0.98}{z-0.64}$$

$D(z)$ 在 z 平面上的极点 $z_p^0 = 0.64$,零点 $z_z^0 = 0.98$,将它们映射到 s 平面上可得

$$s_p = \frac{1}{T_s}\ln(z_p^0) = \frac{1}{0.04}\ln(0.64) = -11.16$$

$$s_z = \frac{1}{T_s}\ln(z_z^0) = \frac{1}{0.04}\ln(0.98) = -0.505$$

$T_s' = 0.1\text{s}$ 时,将 s_p,s_z 再映射到 z 平面上可得

$$z_p' = e^{T_s' s_p} = e^{-0.1\times 11.16} = 0.3277$$

$$z_z' = e^{T_s' s_z} = e^{-0.1\times 0.505} = 0.9508$$

$$D'(z) = k_z'\frac{z-0.9508}{z-0.3277}$$

根据稳态值相等的原则确定 k_z':按根匹配法,必须要求两者非零有限稳态值相等。

$D(z)$ 在单位阶跃信号作用下有稳态值,根据终值定理

$$u(\infty) = \lim_{z\to 1}\left\{\frac{z-1}{z}D(z)\frac{z}{z-1}\right\} = \lim_{z\to 1}\left\{\frac{z-1}{z}2.62\frac{z-0.98}{z-0.64}\frac{z}{z-1}\right\} = 0.14556$$

$D'(z)$ 在单位阶跃信号作用下也应有同样的终值,即

$$\lim_{z\to 1}k_z'\frac{z-0.9508}{z-0.3277} = 0.14556$$

$$k_z' = 1.989$$

从而可得

$$D'(z) = 1.989\frac{z-0.9508}{z-0.3277}$$

■

4.5.3 纯延迟环节的仿真模型

设纯延迟环节如图 4.54 所示，传递函数为

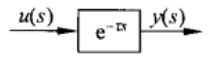


图 4.54 纯延迟环节

$$G(s)=\frac{y(s)}{u(s)}=\mathrm{e}^{-\tau s}$$

其中 $y(s)$与 $u(s)$分别为该环节输出、输入的拉普拉斯变换，τ 为延迟时间。设仿真步距为 T，且

$$\tau/T=C_0+C_2$$

式中 C_0 为整数部分；C_2 为小数部分，则有

$$G(s)=\mathrm{e}^{-(C_0+C_2)Ts}$$

取 $G(s)$的 z 变换得

$$G(z)=\frac{y(z)}{u(z)}=z^{-(C_0+C_2)}$$

若将 $G(z)$反变换，可得差分方程如下

$$y(k)=u[k-(C_0+C_2)] \tag{4.95}$$

下面说明基于式(4.95)模型的仿真方法，先假设比较简单的情况，即 $C_2=0$，也就是延迟时间为仿真步距 T 的整数倍，所以得到

$$y(k)=u(k-C_0) \tag{4.96}$$

式(4.96)所示模型在仿真程序中需这样处理，即开辟 C_0+1 个内存单元预先存放 $u(k)$及以前时刻的值。设内存单元为

$$\begin{array}{cccccc} M(1) & M(2) & \cdots & M(C_0) & M(C_0+1) \\ \uparrow & \uparrow & & \uparrow & \uparrow \\ u(k-C_0) & u(k-C_0+1) & \cdots & u(k-1) & u(k) \end{array}$$

将当前计算出来的 $u(k)$存放在 C_0+1 号单元，而 $y(k)$则从第 1 号单元去取。其步骤是：

(1) 当前计算出来的 $u(k)$存入 $M(C_0+1)$；

(2) 从 $M(1)$取出 $u(k-C_0)$，即为 $y(k)$；

(3) 为计算下一步，需把各内存单元中的数据按单元号减少方向依次传递即 $M(2)\to M(1)$，$M(3)\to M(2)$，…，$M(C_0+1)\to M(C_0)$；

(4) 返回步骤(1)。

所以，每仿真一次进行一次同样的操作，总是按“存入-取出-平移”的顺序由程序实现延迟 C_0T 的功能。

如果数字控制器和被控对象中均含有纯延迟环节，且采样周期 T_s 与仿真步距 T 不等，除了各自开辟一个数据区外，还必须按各自步距进行数据处理。

如果 $C_2\neq 0$，则因 $C_0<C_0+C_2<C_0+1$，所以，$y(k)$的值应在 $u(k-C_0)$和 $u[k-(C_0+1)]$两个数值之间，一般可用以下线性插补公式来求得 $y(k)$，即

$$y(k)=(1-C_2)u(k-C_0)+C_2u[k-(C_0+1)] \tag{4.97}$$

因此，当 $C_2\neq 0$ 时，还需增设内存单元用于存放 $u[k-(C_0+1)]$，则此时需有 C_0+2 个内存单元，其中分别存放以下数据

$$\begin{array}{cccccc} M(0) & M(1) & \cdots & M(C_0) & M(C_0+1) \\ \uparrow & \uparrow & & \uparrow & \uparrow \\ u[k-(C_0+1)] & u(k-C_0) & \cdots & u(k-1) & u(k) \end{array}$$

其步骤是：

(1) 当前计算出来的 $u(k)$存入 $M(C_0+1)$；

(2) 取值，根据式(4.97)所示线性插补公式，从 $M(0)$和 $M(1)$号计算 $y(k)$；

(3) 将 C_0+2 个单元顺序平移一次；

(4) 返回步骤(1)。

4.5.4 采样控制系统仿真举例

本节将通过一个实例，讨论采样控制系统仿真时的一些特点及处理方法。

设采样控制系统的结构如图 4.55 所示。其中被控对象为一阶惯性加纯延迟 $G(s)=\dfrac{k_0}{T_0s+1}e^{-\tau s}$，采用零阶保持器 $G_h(s)=\dfrac{1-e^{-T_s s}}{s}$，$T_s$ 为采样周期。现需要设计数字控制器，使该系统在单位阶跃信号作用下为最小拍系统。

根据最小拍系统的设计思想，该系统在单位阶跃函数作用下，闭环传递函数为

$$W_b(z)=z^{-(\beta+1)}$$

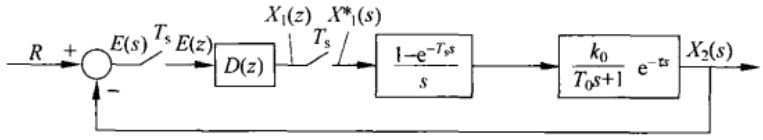


图 4.55 具有纯延迟的控制系统框图

其中 $\beta=\tau/T_s$，并设 β 为整数。又由于

$$W(s)=G_h(s)G(s)=\frac{k_0(1-e^{-T_s s})e^{\tau s}}{s(T_0s+1)}$$

$$W(z)=\mathcal{Z}\{W(s)\}=\mathcal{Z}\left\{\frac{k_0(1-e^{-T_s s})e^{-\tau s}}{s(T_0s+1)}\right\}=\mathcal{Z}\left\{\frac{k_0(1-e^{-T_s s})e^{-\beta T_s s}}{s(T_0s+1)}\right\}$$

$$=k_0(1-e^{-T_s/T_0})\frac{z^{-(\beta+1)}}{1-e^{-T_s/T_0}z^{-1}}$$

从而可得数字控制器 $D(z)$的 z 传递函数为

$$D(z)=\frac{X_1(z)}{E(z)}=\frac{W_b(z)}{(1-W_b(z))W(z)}=\frac{1-e^{-T_s/T_0}z^{-1}}{k_0(1-e^{-T_s/T_0})(1-z^{-(\beta+1)})}$$

取 z 反变换，并写成差分方程的形式

$$x_1(k+1)-x_1(k+1-\beta-1)=\frac{1}{k_0(1-e^{-T_s/T_0})}[e(k+1)-e^{-T_s/T_0}e(k)]$$

若记 $e^{-T_s/T_0}=\alpha$，可简写如下

$$x_1(k+1)=x_1(k-\beta)+\frac{1}{k_0(1-\alpha)}[e(k+1)-\alpha e(k)]$$

这是一个 β 阶的差分方程。

对本系统进行仿真时，应分两部分进行，这是因为仿真步距 T 并不一定与 T_s 相等(T_s 为采样间隔)。对离散的数字控制器部分每隔一个采样间隔 T_s 应计算一次，而在采样间隔之间，要根据信号重构器的特征加以重构。信号重构器的输出为连续部分的输入。对连续部分，为使仿真精度得到保证，取 $T_s=NT$，$N>1$，且一般为整数，以保证两种采样开关同

步。连续部分每隔时间 T 计算一次，计算 N 次后，其输出 $x_2(k)$ 才能用来与 $R(k)$ 进行比较以输入给 $D(z)$。在连续部分计算 N 次以内，其输入由保持器的输出提供，而保持器的输入则保持不变。特别是，当采用零阶保持器时，在 NT 内均保持不变。

另外，由于本系统被控对象存在纯延迟环节，该环节可视为位于惯性环节输出之后，也可视为位于惯性环节输入之前，仿真时可灵活考虑。现考虑纯延迟环节位于惯性环节之前，那么保持器的输出经延迟 β 个采样周期后才输入到惯性环节。即

$$C(k+1) = u(k+1-\beta)$$

此时惯性环节的差分方程为

$$x_2^n(k+1) = \mathrm{e}^{-T/T_0} x_2^n(k) + k_0(1-\mathrm{e}^{-T/T_0})C^n(k)$$

说明：

(1) 本差分模型中未考虑控制量 $C(k)$ 的变化。这是因为零阶保持器的原因。

(2) 差分模型中上标 n 表示第 n 个采样周期，而 k 表示在第 n 个采样间隔内的第 k 次计算 ($k=1,2,\cdots,N$)。从而该采样控制系统仿真程序框图如图 4.56 所示。利用该仿真程序，当 $T_0=1\mathrm{s}$，$k_0=1$，$T_s=0.5\mathrm{s}$，$\tau=1\mathrm{s}$ 时，取仿真步距 $T=0.05\mathrm{s}$，$R=10\cdot 1(t)$，仿真结果如表 4.15 所示。

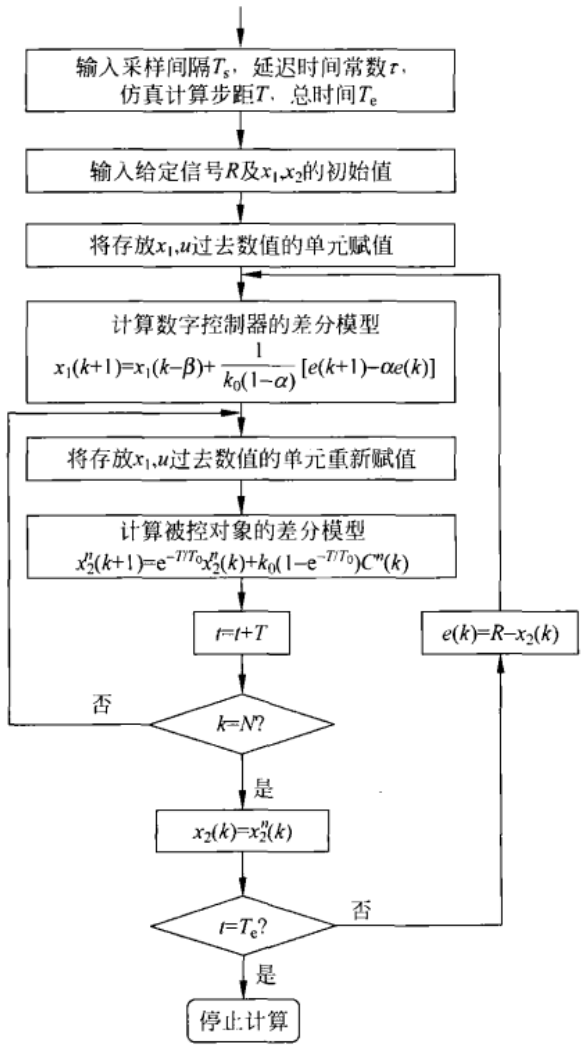


图 4.56 最小拍系统仿真程序框图

表 4.15 采样周期 $T_s=0.5s$ 仿真结果

t	x_1	x_2
0	25.414 9	0
0.25	25.414 9	0
0.50	10	0
0.75	10	0
1.0	10	0
1.25	10	5.621 7
1.5	10	10
1.75	10	10
2.0	10	10
2.5	10	10
2.75	10	10

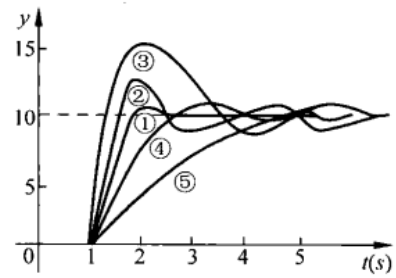
由于 $\beta=\tau/T_s=1/0.5=2$，系统应该在第三个采样周期完成，达到稳定。仿真结果证明了这一点。

从控制理论可知，按最小拍设计的系统适应性差，对参数变化的敏感性大，输出存在纹波。仿真结果也很好地证明了这一点。例如，当被控系统中的参数 k_0 变化到 k_1 时，若数字控制器中结构和参数保持不变，即

$$D(z)=\frac{1-az^{-1}}{k_0(1-a)(1-z^{-(\beta+1)})}$$

其中 k_0 为原设计时被控对象的放大倍数。若 $k_1<k_0$，则系统输出出现超调，过渡过程将加长。若 $k_1>k_0$，系统虽无明显超调，但不是最小拍系统，过渡过程也加长。以上情况的仿真结果如图 4.57 所示。

正因为最小拍系统有上述缺点，这种方法很少直接使用到工程上。但是最小拍系统的设计思想仍然在许多设计方法中得到应用。



①表示$k_1=k_0$；②表示$k_1=0.85k_0$；
③表示$k_1=0.65k_0$；④表示$k_1=1.2k_0$；
⑤表示$k_1=1.5k_0$

图 4.57 参数变化对最小拍系统的影响

4.6 小结

本章首先介绍了离散相似法的原理，分别讨论了由状态方程和传递函数如何得到离散模型，即时域离散相似法和频域离散相似法。在此基础上讨论了两类典型应用。

离散相似法是基于现代控制理论提出的，它将一个连续系统进行离散化处理，得到与它等价的离散模型。为了保证两者等价，在离散化过程中，虚拟采样开关的个数、位置、采样周期等必须仔细设计。

离散相似法最大的优点是在仿真过程中无需进行迭代，这样每一步的计算时间开销较少，从而可获得较快的仿真速度。仿真模型的稳定性仅依赖于仿真步长，其缺点是在系统较复杂时不易得到等价的离散模型，特别是，对于一般非线性系统，由于现代控制理论的局限性，其离散模型几乎不可能获得，因而，其应用范围受到一定限制。

本章以典型案例的方式讨论了离散相似法的实际应用，一方面让读者加深离散相似法

用于对连续系统仿真的理解；另一方面，或者更为重要的方面是，当用仿真技术解决实际问题时，模型是仿真的基础，首先必须建立满足仿真目标要求的系统模型，然而再根据模型的特点选择仿真方法。实际上，该系统也可以采用经典的数值积分法进行仿真建模，有兴趣的读者不妨采用第 5 章介绍的 MATLAB 软件进行仿真，并讨论两种仿真建模方法的优劣。

习题

1. 试求$\frac{s}{s+1}$在作用函数 $u(t)$作用下，输出 $y(t)$的差分方程(采用虚拟采样开关及零阶保持器)。

2. 用根匹配法求 $G(s)=\frac{1}{(s+1)(s+2)}$的差分模型(加单位阶跃输入)。

3. 试用 C 语言编制一段程序实现双线性替换法。

4. 试比较 $G(s)=\frac{K}{s}$在 $u(t)=Rt$ 的作用下(R 为常数)，下列方法所得的差分模型：

(1) 欧拉法；　　(2) 二阶龙格-库塔法；

(3) 一阶保持器加虚拟采样开关；　　(4) 三角保持器加虚拟采样开关。

5. 用 $s=\frac{z-1}{Tz}$的关系对 $G(s)$进行替换，试问

(1) 若 $G(s)$是稳定的，$G(z)$也是稳定的吗？

(2) 若 $G(z)$是稳定的，$G(s)$也一定稳定吗？

6. 有一闭环控制系统如图 4.58 所示，试求$\frac{1}{s(s+2)}$的$\boldsymbol{\Phi}(T)$及$\boldsymbol{\Phi}_{\mathrm{m}}(T)$，并列出求解 $y(t)$的差分方程(提示：在×处断开加虚拟采样开关和零阶信号重构器)。

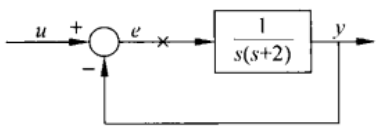


图 4.58　习题 6 系统图示

7. 确定图 4.59 中两个模型等价的 λ,γ 的值：

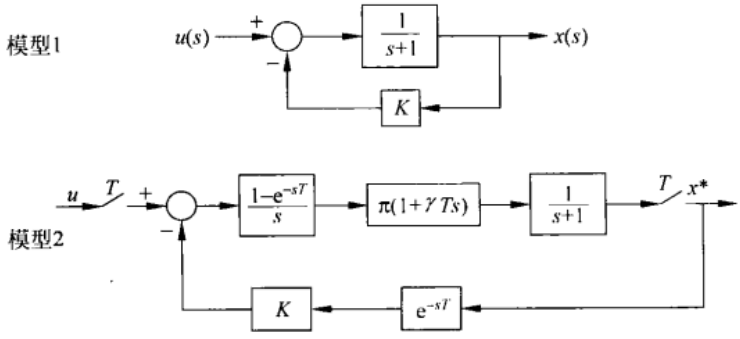


图 4.59　习题 7 系统模型和离散化后的等价模型

8. 试求$\frac{Y(s)}{U(s)}=G(s)=\frac{4}{s^2+5s+6}$的$\boldsymbol{\Phi}(T)$和$\boldsymbol{\Phi}_m(T)$，并列出求解$y(t)$的差分方程，其中$u(t)=R\cdot 1(t)$，$R$为幅值。

9. 用增广矩阵法求$\frac{Y(s)}{U(s)}=\frac{1}{s(s+1)}$的增广方程，其中$u(t)=a(1-e^{-t})$。

10. 试用C语言编制一个函数，可对图4.60所示的非线性环节进行仿真。

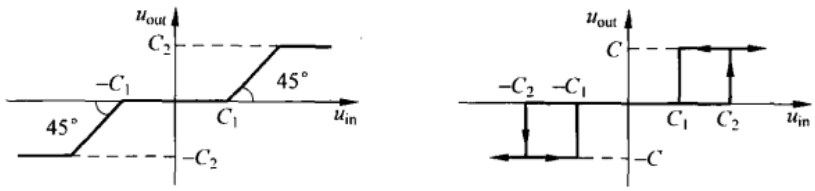


图4.60 习题10非线性环节图示

11. 已知$D(s)$为PID调节器传递函数

$$D(s)=\frac{u(s)}{e(s)}=K_p\frac{1+T_is+T_dT_is^2}{T_is}$$

其中T_i为积分时间常数；T_d为微分时间常数；K_p为比例系数。求该调节器的差分方程。

12. 已知采样系统如图4.61所示，其中$D(z)=\frac{K_v+K_p(1-z^{-1})}{1-z^{-1}}$，状态方程$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}0 & 0\\ 1 & -1/\tau\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix}u$，输出方程$y=\frac{1}{\tau}x_2$。若采用步长$T$对连续部分仿真，且$T_s=LT$（$L$为正整数），试求连续部分的差分模型，试画出整个系统的仿真程序流程图。

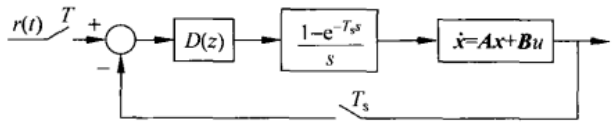


图4.61 习题12所示系统的模型

参考文献

[1] Smith J M. Mathematical Modeling and Digital Simulation for Engineers and Scientists, New York: John Wiley & Sons Inc., 1977

[2] 冯康等. 数值计算方法. 北京：国防工业出版社，1978

[3] 科恩G A，韦特J V. 李仰东译. 连续系统数字仿真. 北京：科学出版社，1981

[4] 熊光楞等. 连续系统仿真与离散事件系统仿真. 北京：清华大学出版社，1991

[5] 刘植桢等. 计算机控制. 北京：清华大学出版社，1981

[6] 佛特曼海兹著. 吕林等译. 线性控制系统引论(An Introduction to Linear Control System). 北京：机械工业出版社，1983

[7] 高龙等. 电气传动系统中两维观测器的工程设计. 自动化学报，1982，4

[8] 肖田元等. 克服大型雷达伺服系统机械谐振的一种新的设计方法及仿真. 信息与控制，1986，3

第5章
集中参数连续系统仿真语言

仿真建模方法学提供了将连续系统模型转换成便于计算机执行的离散模型的方法。然而，当对一个系统进行仿真时，必须为用户提供仿真环境，包括交互式建立系统模型的环境、定义仿真方法的环境、定义试验条件和仿真运行的环境，以及仿真输出分析的环境等，这就是仿真运行问题。

仿真运行依赖于仿真支撑系统，经过多年的发展，典型的仿真支撑系统是仿真语言。本章首先介绍集中参数连续系统仿真语言的发展沿革，然后重点介绍最常见的连续系统仿真语言 MATLAB，因为 MATLAB 代表了集中参数连续系统现代仿真语言技术实现和运行环境的特点。MATLAB 使用得如此广泛，以至于不只是仿真领域，其他工程领域也把它作为一个基本工具。

本章不是从使用 MATLAB 的角度来介绍，而是从仿真工程师学习和掌握通用仿真技术的角度讨论。关于如何使用 MATLAB，读者应参考相关的详细资料。

5.1 发展沿革

仿真语言是仿真研究中基于通用计算机语言而又比其更高级的软件系统。仿真语言为仿真研究人员提供了专门用于建模、仿真试验和仿真结果统计、分析、显示的程序语句，研究人员可以不必深入掌握通用计算机高级语言的编程细节和技巧，而用比较习惯或容易掌握的表达方式来描述仿真模型，把主要精力集中在仿真研究上。

早期的集中参数连续系统仿真语言都属于框图语言系统。1955 年出现了第一个框图式仿真语言——数字模拟仿真语言 DAS。1958 年，Roparker 和 Stein 发表了 ASTRAL (analog schematic translator to algebra language)。1961 年，贝尔实验室发展了框图程序 BLODC(block diagram compiler)；1962 年，Herley Skiles 和 Rldsat 推出了 DYSAC (digitally simulated analog computer)；同年，Solver 和 Knadyson 发表了 PARTNER

(proof of analog results through numerical equivalent results)；1963 年 Petersonn Sansom 发表了 MIDAS(modified integration digital analog simulator)；1964 年，Brennan Sano 推出了 PACTOLUS，这是第一个用于小型计算机的数字仿真语言，它具有一定的人机对话功能；1967 年模块式连续系统建模语言 Block CSMP(block continuous system modeling program)由 IBM 推出，1969 年出现了 Block CSMP 的一种特殊版本 MOBSSL，允许用户在图形显示屏幕上用光笔来选择和互连各个框图算子符号。

在集中参数连续系统仿真语言发展过程中，几乎同时产生了面向方程的仿真语言。面向方程仿真语言系统可使用户很方便地直接以状态方程的形式描述仿真模型，再通过编译程序自动地翻译成 FORTRAN 程序，然后进行编译、装入和执行，得出变量的时间历程，通过一些简单的输出命令就能显示、绘图和打印出来。将 FORTRAN 语言作为宿主语言，不仅能借用现有的 FORTRAN 编译程序，而且还可以使有经验的用户在仿真研究中充分利用 FORTRAN 函数和过程。在多次运行仿真研究中可用 FORTRAN 控制程序来进行控制。对于没有 FORTRAN 语言程序设计知识的用户，也很容易掌握这些高级语言的特点，很快就能学会求解各种微分方程。典型的是 1959 年出现的系统动力学建模语言 DYNAMO。1965 年，美国成功研制了第一个获得广泛应用的面向方程的仿真语言 MIMIC，1966 年，美国的 Syn 和 Lineberger 推出了另一个面向方程的仿真语言 DSL/90(digital simulation language)，它与 FORTRAN 兼容；1967 年美国计算机仿真学会总结了面向方程的仿真语言的特点，并加以标准化，提出一种兼有框图表示功能的面向方程的仿真语言——连续系统仿真语言(CSSL)，成为连续系统仿真语言的规范。此后出现了许多符合 CSSL 规范的语言。

之后，上述两种形式的语言逐渐融合，并相继推出了多种仿真语言，如美国宾州大学研制了面向结构图的 CSS(continuous system simulation)；荷兰台尔福特大学研制了面向结构图的 PSI；1967 年，美国的 Arizona 大学开始了针对连续系统的仿真语言 DARE(differential analyzer replacement)研制，先后推出了 DARE-K、DARE-F、DARE-P(1974 年)和 DARE-11(1976 年)。

1976 年，美国的 Mitchell 和 Gauthier Associates(MGA)开发了 ACSL(advanced continuous simulation language)；1977 年，GASP(general activity simulation program)Ⅳ和 GSL(general simulation language)问世；此外，美国的 Pritsker & Associates Inc. 研制了针对连续离散混合系统的仿真语言 SLAM；美国的 ISI(Integrated System, Inc)公司开发了 MATRIXx。

20 世纪 70 年代末，Orën 提出先进的仿真方法学，即建模-试验-分析的概念框架，为一体化建模/仿真环境研制奠定了基础。1981 年，美国仿真学会及国际数学与计算机学会对 CSSL67 进行修改，提出了 CSSL 新的规则，有 4 项较大的改进，包括模型与试验分离描述、允许使用子模型、增加间断点特性以及并行程序段等。与此同时，MATLAB 问世，并逐步发展成为最流行的连续系统仿真语言。

1983 年，美国陆军与美国高级研究计划局共同制定了仿真器网络 SIMNET(SIMulation NETworking)计划，标志着分布交互仿真环境研究的开始。此后，分布交互仿真环境的发展经历了 SIMNET、DIS(distributed interactive simulation)、ALSP(aggregate level simulation protocol)和 HLA(high level architecture)四个主要阶段，其体系结构和技术标准对现代建模仿真环境产生了重要影响。1984 年，加拿大 GEST 仿真软件系统研制成功，它通过数据库系统，将建模、仿真执行及输出分析等阶段组成一个有机的整体；对现代建模

仿真环境产生或将要产生重要影响的体系结构和标准规范当然不止于分布交互仿真技术。

2001年冬季仿真会议上，Kilgore R A展示了一个名为OpenSML(open simulation modeling language)的开源项目，其目标是建立一种具有可读性和扩展性的模块化、通用化的建模仿真标准。SML基于XML标准，独立于特定的实现语言，并试图支持各种实现语言与SML之间的转换。目前，SML已经提出了关于仿真模型、仿真试验和仿真结果的一系列标准草案。2002年，美国波音公司制定了基于可扩展标记语言XML标准(extensible markup language)的仿真参考标记语言SRML规范(simulation reference markup language)，以应对模型和仿真开发中面临的不断增长的集成和重用的复杂性。SRML试图概括建模仿真的基本元素和活动，对各种仿真模型进行规范化表达，并将仿真模型与仿真运行相分离，以支持面向不同领域的应用扩展和基于XML标准的模型集成。目前，SRML还处于发展阶段，需要与建模仿真的理论、方法、技术和平台充分结合。

近年来，国际标准化组织提出的模型驱动体系结构MDA(model driven architecture)在计算机信息领域引起广泛重视，并将对建模仿真技术的未来发展产生重要影响。MDA试图采用统一建模语言(UML)等抽象语言对软件的设计模型进行统一描述，并在一系列基础构件和商用技术标准的支持下，自动生成C++代码、Java代码、C#代码、测试框架、整合代码、Java对象、.Net对象、部署脚本乃至软件文档，而且可以将其映射到J2EE、.Net、CORBA等平台，以最大限度地提升软件的开发效率和可重用程度。借鉴MDA思想，可以建立具有较高平台无关性的仿真语言及仿真标准，用以支持仿真模型的设计、存储、交换、执行、可视化等功能的实现，并促进仿真模型组合、重用等性能的提高。

5.2 MATLAB简介

5.2.1 MATLAB发展历程

MATLAB的产生是与数学计算联系在一起的。1980年美国新墨西哥州大学计算机系主任Cleve Moler在给学生讲授线性代数课程时，发现学生在高级语言编程上花费很多时间，于是着手编写供学生使用的FORTRAN子程序库接口程序，他将这个接口程序取名为MATLAB(即matrix laboratory的前3个字母的组合，意为“矩阵实验室”)。这个程序获得很大的成功，受到学生的广泛欢迎。

经过在该大学试用以后，于1984年成立了Math Works公司，推出了采用C语言内核的正式版本。该软件集中了日常数学处理中的各种功能，包括高效的数值计算、矩阵运算、信号处理和图形生成等功能。在MATLAB环境下，用户可以集成地进行程序设计、数值计算、图形绘制、输入输出、文件管理等各项操作。后来版本又增加了丰富的图形图像处理及多媒体功能，使得MATLAB的应用越来越广泛。

Math Works公司正式推出MATLAB后，于1992年推出了具有划时代意义的MATLAB 4.0版；1999年推出的MATLAB 5.3版在很多方面进一步改进了MATLAB的功能，随之推出的Simulink 3.0也达到了很高的档次；2000年10月推出的MATLAB 6.0版，在操作界面上有了很大的改观，同时还给出了程序发布窗口、历史信息窗口和变量管理

窗口等，为用户提供了极大的方便；与之相匹配的 Simulink 4.0 版的新功能也特别引人注目。2001 年 6 月，MATLAB 6.1 版及 Simulink 4.1 版问世，功能已经十分强大，其虚拟显示工具箱更给仿真结果三维视景下显示带来了新的解决方案；2002 年 6 月，又推出了 MATLAB 6.5 版及 Simulink 5.0 版，在计算方法、图形功能、用户界面设计、编程手段和工具等方面都有重大改进。2004 年 7 月，Math Works 公司推出了最新的 MATLAB 7.0 版，其中集成了最新的 MATLAB 7 编译器、Simulink 6.0 仿真软件以及很多工具箱。这一版本增加了很多新的功能和特性，内容十分丰富。

5.2.2 MATLAB 系统构成

MATLAB 系统由 MATLAB 开发环境、MATLAB 数学函数库、MATLAB 语言、MATLAB 图形处理系统和 MATLAB 应用程序接口(API)五大部分构成。

(1) MATLAB 开发环境

MATLAB 开发环境是一套方便用户使用 MATLAB 函数和文件的工具集，其中许多工具是图形化用户接口。它是一个集成化的工作空间。可以让用户输入输出(I/O)数据，并提供了 m 文件编辑调试器、MATLAB 工作空间和在线帮助文档。

(2) MATLAB 数学函数库

MATLAB 数学函数库包括大量的计算算法，从基本运算，如加法、正弦等到复杂算法，如矩阵求逆、贝塞尔函数、快速傅里叶变换等。

(3) MATLAB 语言

MATLAB 语言是一个高级的基于矩阵/数组的语言，有程序流程控制、函数、数据结构、输入输出和面向对象编程等特色。用户既可以用它来快速编写简单的程序，也可以用来编写庞大复杂的应用程序。

(4) MATLAB 图形处理系统

图形处理系统使得 MATLAB 能方便地图形化显示向量和矩阵，而且能对图形添加标注和打印。它包括强有力的二维、三维图形函数，图像处理和动画显示等函数。

(5) MATLAB 应用程序接口(API)

MATLAB 应用程序接口是一个使 MATLAB 语言能与 C、FORTRAN 等其他高级编程语言进行交互的函数库，该函数库的函数通过调用动态链接库(DLL)实现与 MATLAB 文件的数据交换，其主要功能包括在 MATLAB 中调用 C 和 FORTRAN 程序，以及在 MATLAB 与其他应用程序间建立客户/服务器关系。

5.2.3 MATLAB 工具箱

MATLAB 拥有一个专用的家族产品，用于解决不同领域的问题，称为工具箱(TOOLBOX)，工具箱用于 MATLAB 的计算和画图，通常是 m 文件和高级 MATLAB 语言的集合，以使用户可以方便地修改函数和源代码，或增加新的函数。用户还可以很方便地结合使用不同的工具箱中的技术来设计针对某个问题的用户解决方案。MATLAB 每年都会增加一些新的工具箱，所以，在一般情况下，工具箱的列表不是固定不变的。较为常见的

MATLAB 工具箱包括以下几类。

(1) 控制类工具箱

控制系统工具箱(control system toolbox)

系统辨识工具箱(system identification toolbox)

鲁棒控制工具箱(robust control toolbox)

模糊逻辑工具箱(fuzzy logic toolbox)

神经网络工具箱(neural network toolbox)

频域系统辨识工具箱(frequency system identification toolbox)

模型预测控制工具箱(model predictive control toolbox)

多变量频率设计工具箱(multivariable frequency design toolbox)

(2) 应用数学类工具箱

最优工具箱(optimization toolbox)

样条工具箱(spline toolbox)

统计工具箱(statistics toolbox)

偏微分方程工具箱(partial differential equation toolbox)

(3) 信号处理类工具箱

信号处理工具箱(signal processing toolbox)

滤波器设计工具箱(filter design toolbox)

通信工具箱(communication toolbox)

小波分析工具箱(wavelet toolbox)

高阶谱分析工具箱(higher order spectral analysis toolbox)

(4) 其他常用的工具箱

符号数学工具箱(symbolic math toolbox)

虚拟现实工具箱(virtual reality toolbox)

(5) MATLAB 7.0 对以前的版本增加的工具箱

定点工具箱(fixed point toolbox)

遗传算法和直接搜寻工具箱(genetic algorithm and direct search toolbox)

射频工具箱(RF toolbox)

生物信息工具箱(bioinformatics toolbox)

OPC 服务器工具箱(OPC toolbox)

5.2.4 MATLAB 7.0/Simulink 6.0 特点

MATLAB 既是高级计算语言,又是进行数据分析和算法开发的集成开发环境。MATLAB 7.0 在 MATLAB 6.5 的基础上进行了很多改善和提高,Simulink 6.0 在 Simulink 5.0 的基础上也增加了不少新功能,使得这一软件的性能大大提高。

1. MATLAB 7.0 最新特点

MATLAB 7.0 在编程环境、代码效率、数据可视化、数学计算、文件 I/O 等方面进行了

升级，包括以下几方面：

(1) 开发环境方面

① 重新设计的桌面环境，针对多文档界面应用提供了简便的管理和访问方法，允许用户自定义桌面外观、创建常用命令的快捷方式。

② 增强数组编辑器(array editor)和工作空间浏览器(workspace brower)功能，用于数据的显示、编辑和处理。

③ 在当前目录浏览器(current directory brower)工具中，增加代码效率分析、覆盖度分析等功能。

④ M-Lint 编码分析，辅助用户完成程序性能分析，提高程序执行效率。

⑤ 增强 m 文件编辑器(m-editor)，支持多种格式源代码文件可视化编辑，如 C/C++、HTML、Java 等。

(2) 编程方面

① 支持创建嵌套函数(nested function)，提供更灵活的代码模块化方式。

② 匿名函数(anonymous function)功能，支持在命令行或者脚本文件中创建单行函数(single line function)。

③ 支持条件分支断点，可以在条件分支语句中进行程序中断调试。

④ 模块化注释，支持为代码段注释。

(3) 数学运算方面

① 支持整数算术运算。

② 支持单精度数据类型运算，包括基本算术运算、线性代数、FFT 等。

③ 使用强大的计算算法包 Qhull 2002.1，提供更丰富的算法支持。

④ linsolve 函数用于进行线性代数方程求解。

⑤ ODE 求解器能够处理隐性微分方程组以及多点边界问题。

(4) 图形和 3D 可视化方面

① 新图形窗体界面。

② 直接从窗体图形生成 m 代码，可以完成用户自定义绘图。

③ 增强图形窗体注释。

④ 数据侦测工具(data exploration tools)，提供丰富的数据观测手段。

⑤ 自定义图形对象，提供丰富的图形显示能力。

⑥ GUIDE 新增对用户界面面板和 ActiveX 控件的支持。

⑦ 增强句柄图形对象支持完整的 TeX 和 LaTex 字符集。

(5) 文件 I/O 函数和外部接口方面

① 新增文件 I/O 函数，支持读取任意格式文本数据文件，并且支持写入 Excel 和 HDF5 格式数据文件。

② 具有压缩功能的 MAT 文件格式，支持快速数据文件 I/O 能力。

③ javaaddpath 函数，无须重新启动 MATLAB 即可完成 Java 类的加载、删除等功能。

④ 支持 COM、服务器事件以及 VBS。

⑤ 支持 SOAP，使用网络服务。

⑥ 支持 FTP 对象，直接访问 FTP 服务器。

⑦ 支持 Unicode 编码形式，增强 MAT 文件字符集。

(6) 性能与系统平台支持方面

① JIT 加速器支持所有数值数据类型。

② Windows XP 系统下支持 3GB 内存访问。

2. Simulink 6.0 最新特点

Simulink 于 20 世纪 90 年代初由 Math Works 公司开发,是 MATLAB 环境下对动态系统进行建模、仿真和分析的一个软件包。从字面上看,"Simulink"一词有两层含义,"simu"表明它可用于系统仿真,"link"表明它能进行系统连接。在该软件环境下,用户可以在屏幕上调用现成的模块,并将它们适当连接起来以构成系统的模型,即所谓的可视化建模。建模以后,以该模型为对象运行 Simulink 中的仿真程序,可以对模型进行仿真,并可以随时观察仿真结果和干预仿真过程。Simulink 由于功能强大、使用简单方便,已成为应用最广泛的动态系统仿真软件。

Simulink 较为流行的版本有:与 MATLAB 5.2 配用的 Simulink 2.2、与 MATLAB 5.3 配用的 Simulink 3.0、与 MATLAB 6.0 配用的 Simulink 4.0、与 MATLAB 6.5 配用的 Simulink 5.0 以及与 MATLAB 7.0 配用的 Simulink 6.0。Simulink 既适用于线性系统,也适用于非线性系统;既适用于连续系统,也适用于离散系统和连续与离散混合系统;既适用于定常系统,也适用于时变系统。

Simulink 是一个交互式动态系统建模、仿真和分析的图形环境,是一个进行基于模型的嵌入式系统开发的基础开发环境。Simulink 可以针对控制系统、信号处理和通信系统等进行系统建模、仿真、分析等工作。Simulink 6.0 改善了性能,并且针对大规模系统开发进行了性能优化,主要包括以下几方面。

(1) 大系统建模方面

① 支持将大系统模型分为不同的文件,每个文件为独立的系统模型。

② 支持系统不同模型文件独立仿真测试。

③ 增强系统集成手段,支持配置管理和版本控制软件。

④ 递增式模型加载与代码生成功能。

⑤ 针对大模型系统提高运行速度和效率。

⑥ 模型工作空间(model workspace),每个模型都用于独立的工作空间进行参数管理和数据处理。

⑦ 增强总线支持。

(2) Simulink 和 Stateflow 方面

① 统一的模型浏览器(model explorer),用于浏览、维护、配置、搜索、定义所有模型中相关的参数、数据对象和属性。

② 统一的仿真和代码生成选项。

③ 支持创建并保存多种仿真和代码生成选项。

④ 数据管理和可视化。

⑤ 新增数据对象属性。

⑥ 可选数据记录增加测试点,无须在模型中增加额外的模块。

⑦ I/O 管理,可以将必要的信号源和信宿连接到模型而不需要增加模块。

(3) MATLAB 支持方面

① 使用嵌入式 MATLAB(embedded MATLAB)功能引入算法并支持 C 代码生成。

② 增强 M 语言 S 函数的支持。

5.3 MATLAB 的模型设计

Simulink 提供图形用户界面,用户可以用鼠标操作,从模块库中调用标准模块,将它们适当地连接起来以构成动态系统模型,并且用各模块的参数对话框为系统中各模块设置参数。当各模块的参数设置完成之后,即建立起该系统的模型。如果对某一模块没有设置参数,那就意味着使用 Simulink 预先为该模块设置的默认参数值作为该模块的参数。

Simulink 模块库内容十分丰富,除包括输入信源模块库(sources)、输出信宿模块库(sinks)、连续系统模块库(continuous)、离散系统模块库(discretes)、数学运算模块库(math operations)等许多标准模块外,用户还可以自定义和创建模块。

系统的模型建立之后,选择仿真参数和数值算法,便可以启动仿真程序对该系统进行仿真,这种操作可以用 Simulink 菜单,也可以用 MATLAB 命令实现。菜单方式对于交互式运行特别方便,而命令方式对于运行一批仿真时很有用。

在仿真过程中,用户可以设置不同的输出方式来观察仿真结果。例如,使用 sink 模块库中的 scope 模块或其他显示模块来观察有关信号的变化曲线,也可以将结果存放在 MATLAB 工作空间中,供以后处理和使用。

根据仿真结果,用户可以调整系统参数,观察分析仿真结果的变化,从而获得更加理想的仿真结果。MATLAB 系统建模技术包括信源、系统和信宿的模型描述形式及使用要点。

5.3.1 线性连续系统 Simulink 描述

Simulink 允许直接用传递函数和状态方程的形式定义连续系统的模型,MATLAB 内置函数库完成模型间的转换。

1. 传递函数模型

设传递函数分子和分母 s 多项式的阶次分别为 n 和 m,Simulink 用向量 $num=[n_0,n_1,\cdots,n_{n+1}]$和 $den=[d_0,d_1,\cdots,d_{m+1}]$分别定义以 s 降幂形式的传递函数的分子和分母的系数。

2. 零极点模型

若线性连续系统的传递函数用零极点的形式给出,Simulink 用变量 gain$=k$ 定义系统增益,分别用向量 $zeros=[z_1,z_2,\cdots,z_n]$与 $poles=[p_1,p_2,\cdots,p_m]$定义传递函数的分子 s 多项式$(s-z_1)(s-z_2)\cdots(s-z_n)$(其阶次为 n)和分母 s 多项式$(s-p_1)(s-p_2)\cdots(s-p_m)$(其阶次为 m)。

3. 状态空间模型

Simulink 用变换矩阵 A,B,C,D 直接定义如下形式的系统

$$\dot{X}(t) = AX(t) + BU(t) \quad Y(t) = CX(t) + DU(t)$$

这三类模型相互转换的内置函数如下：

$[zeros, poles, k] = tf2zp(num, den)$——将一般传递函数表达式转换为零极点表达式；

$[A, B, C, D] = tf2ss(num, den)$——将一般传递函数表达式转换为状态空间描述；

$[num, den] = zp2tf(zeros, poles, k)$——将零极点表达式转换为一般传递函数表达式；

$[A, B, C, D] = zp2ss(zeros, poles, k)$——将零极点表达式转换为状态空间描述；

$[zeros, poles, k] = ss2zp[A, B, C, D]$——将状态空间描述转换为零极点表达式；

$[num, den] = ss2tf[A, B, C, D]$——将状态空间描述转换为一般传递函数表达式。

5.3.2 线性离散时间系统 Simulink 描述

与线性连续系统类似，对线性离散时间系统，Simulink 允许用滤波器模型、零极点模型和状态空间模型描述，分别是：

1. 滤波器模型

以 z^{-1} 升幂形式定义的传递函数，分子和分母 z^{-1} 多项式的阶次分别为 n 和 m，Simulink 用向量 $num=[n_0, n_1, \cdots, n_{n+1}]$ 和 $den=[d_0, d_1, \cdots, d_{m+1}]$ 分别定义。

z 传递函数模型：设 z 传递函数分子和分母 z 多项式的阶次分别为 n 和 m，Simulink 用向量 $num=[n_0, n_1, \cdots, n_{n+1}]$ 和 $den=[d_0, d_1, \cdots, d_{m+1}]$ 分别定义以 z 降幂形式的传递函数的分子和分母的系数。

2. 零极点模型

若线性连续系统的 z 传递函数用零极点的形式给出，Simulink 用变量 $gain=k$ 定义系统增益，分别用向量 $zeros=[z_1, z_2, \cdots, z_n]$ 和 $poles=[p_1, p_2, \cdots, p_m]$ 定义传递函数的分子 z 多项式 $(z-z_1)(z-z_2)\cdots(z-z_n)$（其阶次为 n）和分母的 z 多项式 $(z-p_1)(z-p_2)\cdots(z-p_m)$（其阶次为 m）。

3. 状态空间模型

Simulink 用变换矩阵 F, G, C, D 直接定义如下形式的系统

$$X(n+1) = FX(n) + GU(n) \quad Y(n) = CX(n) + DU(n)$$

这三类模型相互转换的内置函数如下：

$[zeros, poles, k] = tf2zp(num, den)$——将一般 z 传递函数表达式转换为零极点表达式；

$[F, G, C, D] = tf2ss(num, den)$——将一般 z 传递函数表达式转换为状态空间描述；

$[num, den] = zp2tf(zeros, poles, k)$——将零极点表达式转换为一般 z 传递函数表达式；

$[F, G, C, D] = zp2ss(zeros, poles, k)$——将零极点表达式转换为状态空间描述；

$[zeros, poles, k] = ss2zp[F, G, C, D]$——将状态空间描述转换为零极点表达式；

$[num, den] = ss2tf[F, G, C, D]$——将状态空间描述转换为一般 z 传递函数表达式。

5.3.3 连续-离散时间混合系统 Simulink 描述

连续-离散时间混合系统的模型描述包括连续部分描述、离散时间部分描述，以及连续

信号转换到离散信号(采样)/离散信号转换到连续信号(保持)的描述。

在这类系统中,由于存在采样/保持这样一对功能性元件,其中采样周期和保持器的数学模型是需要专门处理的两个重要方面,以实现连续部分与离散部分的集成。

在 MATLAB 中,这种模型需要用 M 脚本文件来描述。

1. Simulink 高级模型描述

Simulink 高级模型描述的含义是:在视觉上表现为直观的方块结构图,即允许用户基于用图元定义的函数库,以结构图的形式定义被仿真的系统,Simulink 自动地将其转换为一组微分方程或差分方程,并且以扩展名为 MDL 的 ASCII 码文件加以保存。

Simulink 模型一般包含 3 种组件:信源(source)、系统(system)和信宿(sink)。其中信源表示系统的输入(如果没有输入,模型中也可以不定义),信宿表示输出记录方式(如示波器、图形记录仪等)。Simulink 提供的库元素详细情况请参阅用户手册。

Simulink 采用交互式环境,以方便用户建立被仿真的系统结构图。在 MATLAB 安装完成后即可激活,其操作界面如图 5.1 所示。当 Simulink 库浏览器(图 5.2)被启动后,采用鼠标单击相应菜单,可查看模块库、选择库元素、建立模块间的连接关系等,具体的操作过程不再赘述,请参见 MATLAB 使用手册。由以上的介绍可以看出,Simulink 提供的建模功能是十分强大的。

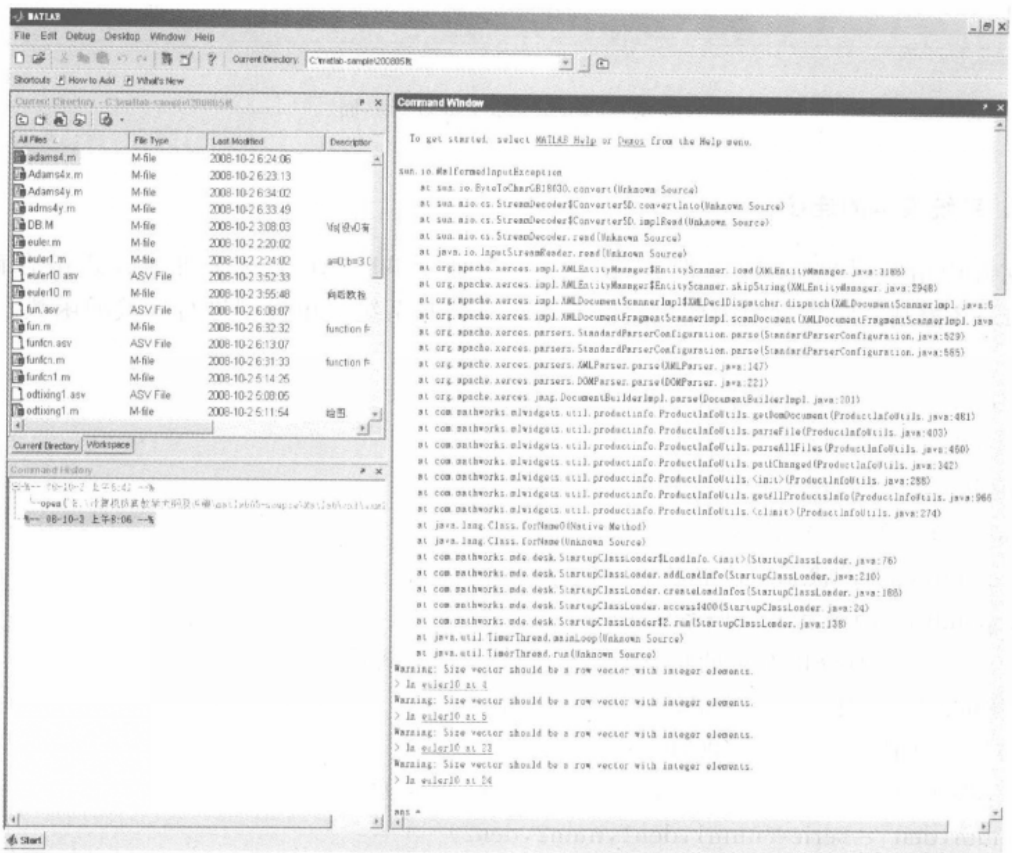


图 5.1 MATLAB 默认的操作界面



图 5.2 Simulink 库浏览器界面

2. 系统模型的连接

实际应用中，控制系统常常由若干个环节通过串联、并联或反馈等方式组合而成，MATLAB 控制系统工具箱因此也提供了大量对控制系统简单模型进行连接的函数。具体如下。

series()：系统的串联连接

parallel()：系统的并联连接

feedback()：系统的反馈连接

cloop()：单位反馈连接

augstate()：将状态增广到状态空间系统的输出中

append()：两个状态空间系统的组合

connect()：对分块对角的状态空间模型按指定方式连接

blkbuild()：把方块图模型转化为状态空间模型

模型相互转换的内置函数如下：

[A,B,C,D]=series(A1,B1,C1,D1,A2,B2,C2,D2)

[num,den]=series(num1,den1,num2,den2)

[A,B,C,D]=parallel(A1,B1,C1,D1,A2,B2,C2,D2)

[num,den]=parallel(num1,den1,num2,den2)

[A,B,C,D]=feedback(A1,B1,C1,D1,A2,B2,C2,D2,sign)
[num,den]=feedback(num1,den1,num2,den2,sign)

3. 系统的离散化

利用 MATLAB 控制系统工具箱的函数可以将连续系统的模型离散化。
c2d()：将状态空间模型由连续形式转为离散形式
c2dm()：连续形式到离散形式的转换(可选不同方法)
c2dt()：连续形式到离散形式的对输入纯时间延迟转换
离散化变换方式选项为：
zoh：假设输入端加一个采样开关和零阶保持器
foh：假设输入端加一个采样开关和一阶保持器
tustin：采用双线性变换(Tustin 算法)方法
prewarp：采用改进的 Tustin 变换方法

5.4 MATLAB 仿真建模技术

MATLAB 仿真建模技术，包括离散时间系统仿真建模(单采样频率与多采样频率)要点、连续系统仿真建模(定步长连续求解器、变步长连续求解器、病态系统求解器)、连续-离散时间混合系统的仿真建模、系统过零技术以及代数环技术等。

仿真建模在 MATLAB 中称为求解器。Simulink 将系统模型及其参数转换成方程，通过求解器转换为仿真模型。这种转换是后台进行的，用户并不能感知到。实际上，当用户选择某一种仿真算法时，就确定了仿真模型的形式。

5.4.1 连续系统仿真建模

对连续系统，MATLAB 采用连续求解器进行仿真。无论是传递函数模型、零极点模型，还是状态空间模型，都将转换为一阶微分方程组，然后转换为仿真模型。

1. 仿真时间

注意这里的时间概念与真实的时间并不一样，只是计算机仿真中对时间的一种表示，比如 10s 的仿真时间，如果采样步长定为 0.1，则需要执行 100 步，若把步长减小，则采样点数增加，那么实际的执行时间就会增加。一般仿真开始时间设为 0，而结束时间视不同的因素而选择。总的说来，执行一次仿真要耗费的时间依赖于很多因素，包括模型的复杂程度、解法器及其步长的选择、计算机时钟的速度等。

2. 仿真步长模式

用户在 Type 后面的第一个下拉选项框中指定仿真的步长选取方式，可供选择的有 variable-step(变步长)和 fixed-step(固定步长)方式。变步长模式可以在仿真的过程中改变

步长，提供误差控制和过零检测。固定步长模式在仿真过程中提供固定的步长，不提供误差控制和过零检测。

用户还可以在第二个下拉选项框中选择对应模式下仿真所采用的算法。固定步长模式解法器有：ode5，ode4，ode3，ode2，ode1 和 discrete，分别说明如下。

ode5：默认值，是 ode45 的固定步长版本，适用于大多数连续系统或离散系统，不适用于刚性系统。

ode4：四阶龙格-库塔法，具有一定的计算精度。

ode3：固定步长的二/三阶龙格-库塔法。

ode2：改进的欧拉法。

ode1：欧拉法。

discrete：是一个实现固定积分步长的解法器，它适合于离散无连续状态的系统。

变步长连续求解器可根据用户指定的计算误差要求自动调整仿真步长，实现仿真过程的误差控制。变步长模式解法器有 ode45，ode23，ode113，ode15s，ode23s，ode23t，ode23tb 和 discrete，分别说明如下。

ode45：默认值，四/五阶龙格-库塔法，适用于大多数连续系统或离散系统，但不适用于刚性(stiff)系统。它是单步解法器，也就是在计算 $y(t_n)$ 时，它仅需要最近处理时刻的结果 $y(t_{n-1})$。一般来说，面对一个仿真问题最好是首先试试 ode45。分别用四阶、五阶龙格-库塔法计算步长终点变量的值，并计算两者之差作为对截断误差大小的估计。

ode23：二/三阶龙格-库塔法，它在误差限要求不高和求解的问题不太难的情况下，可能会比 ode45 更有效。也是一个单步解法器。分别用二阶、三阶龙格-库塔法计算步长终点变量的值，并计算两者之差作为对截断误差大小的估计。如果截断误差估计值大于指定的计算误差，则缩短步长重新计算，直到误差估计值达到指定精度范围，如果截断误差估计值远小于指定的计算误差，则加大下一计算步长。

ode113：是一种阶数可变的解法器，它在误差容许要求严格的情况下通常比 ode45 有效。ode113 是一种多步解法器，也就是在计算当前时刻输出时，它需要以前多个时刻的解。ode113 是变阶的 Adams 法，因此是一种多步法。该方法首先通过前 $(n-1)$ 个点的仿真值拟合一个 $(n-1)$ 阶时间多项式的系数，外推得到当前点的预报值；然后用前 $(n-1)$ 个点的仿真值和当前点的预报值拟合一个 n 阶时间多项式进行校正。校正值与预报值之差用于误差测度，以调整仿真步长。

ode15s：是一种基于数字微分公式的解法器(NDFs)。也是一种多步解法器。适用于刚性系统，当用户估计要解决的问题是比较困难的，或者不能使用 ode45，或者即使使用 ode45 效果也不好，就可以用 ode15s。

ode23s：是一种单步解法器，专门应用于刚性系统，在弱误差允许下的效果好于 ode15s。ode23s 是基于 Rosenbrok 公式建立的定阶单步法，由于阶次不变，它能解决某些 ode15s 所不能有效解决的 Stiff 问题。

ode23t：是梯形规则的一种自由插值实现。这种解法器适用于求解适度 Stiff 的问题而用户又需要一个无数字振荡的解法器的情况。

ode23tb：是 TR-BDF2 的一种实现，TR-BDF2 是具有两个阶段的隐式龙格-库塔公式。

discrete：当 Simulink 检查到模型没有连续状态时使用它。

3. 步长参数

对于变步长模式，用户可以设置最大的和推荐的初始步长参数，默认情况下，步长自动地确定，它由值 auto 表示。

maximum step size(最大步长参数)：它决定了解法器能够使用的最大时间步长，它的默认值为“仿真时间/50”，即整个仿真过程中至少取 50 个取样点，但这样的取法对于仿真时间较长的系统则可能带来取样点过于稀疏，而使仿真结果失真。一般建议对于仿真时间不超过 15s 的采用默认值即可，对于超过 15s 的每秒至少保证 5 个采样点，对于超过 100s 的，每秒至少保证 3 个采样点。

initial step size(初始步长参数)：一般建议使用 auto 默认值即可。

4. 仿真精度的定义(对于变步长模式)

relative tolerance(相对误差)：是指误差相对于状态的值，是一个百分比，默认值为 1e-3，表示状态的计算值要精确到 0.1%。

absolute tolerance(绝对误差)：表示误差值的门限，或者是说在状态值为零的情况下，可以接受的误差。如果它被设成了 auto，那么 Simulink 为每一个状态设置初始绝对误差为 1e-6，表示计算值要精确到 10^{-6}。

5. mode(对于固定步长模式选择)

multitasking：选择这种模式时，当 Simulink 检测到模块间非法的采样速率转换，它会给出错误提示。非法采样速率转换是指两个工作在不同采样速率的模块之间的直接连接。在实时多任务系统中，如果任务之间存在非法采样速率转换，那么就有可能出现一个模块的输出在另一个模块需要时却无法利用的情况。通过检查这种转换，multitasking 将有助于用户建立一个符合现实的多任务系统的有效模型。

使用速率转换模块可以减少模型中的非法速率转换。Simulink 提供了两个这样的模块：unit delay 模块和 zero-order hold 模块。对于从慢速率到快速率的非法转换，可以在慢输出端口和快输入端口插入一个单位延时 unit delay 模块。而对于快速率到慢速率的转换，则可以插入一个零阶采样保持器 zero-order hold。

singletasking：选择这种模式不检查模块间的速率转换，它在建立单任务系统模型时非常有用，这种系统不存在任务同步问题。

auto：Simulink 会根据模型中模块的采样速率是否一致，自动决定切换到 multitasking 或 singletasking。

6. 输出选项

refine output：这个选项可以理解为精细输出，其意义是在仿真输出太稀松时，Simulink 会产生额外的精细输出，这一点就像插值处理一样。用户可以在 refine factor 设置仿真时间步间插入的输出点数。

如要产生更光滑的输出曲线，改变精细因子比减小仿真步长更有效。精细输出只有在变步长模式中才能使用，并且在 ode45 效果最好。

produce additional output：允许用户直接指定产生输出的时间点。一旦选择了该项，

则在它的右边出现一个 output times 编辑框，在这里用户指定额外的仿真输出点，它既可以是一个时间向量，也可以是表达式。与精细因子相比，这个选项会改变仿真的步长。

produce specified output only：意思是让 Simulink 只在指定的时间点上产生输出。为此解法器要调整仿真步长以使之和指定的时间点重合。这个选项在比较不同的仿真时可以确保它们在相同的时间输出。

5.4.2 离散时间系统与连续-离散时间混合系统仿真建模

1. 离散时间系统仿真建模

对离散时间系统，MATLAB 采用离散求解器进行仿真，实际上是对差分方程求解。无论是滤波器模型、z 传递函数、零极点模型，还是状态空间模型，都将转换为差分方程。

对单采样频率的离散时间系统，选择定步长求解器，并且要求仿真步长与系统的采样周期相同。对多采样频率的离散时间系统，如果采用定步长，则所有采样周期必须是仿真步长的整数倍。即使如此，仍可能造成多余的计算。例如，假设系统有两种采样周期，分别为 0.4s 和 0.6s，则仿真步长只能选 0.2s，每隔 0.2s 计算一次，这样在 0.2s、1.0s、1.4s、…进行了不必要的计算。MATLAB 允许对多频率的离散时间系统采用变步长求解器计算，从而提高了仿真效率。总的来说，离散时间系统的仿真建模比较简单，只需要将各种不同形式离散模型转换成差分模型，即可用于仿真。

2. 连续-离散时间混合系统仿真建模

连续-离散时间混合系统的仿真建模主要是连续信号与离散信号之间的匹配问题，Simulink 要求采用变步长连续求解器，通过调整仿真步长以匹配离散信号的采样间隔。

由于混合系统中信号类型不一，Simulink 仿真环境提供了 sample time colors 功能用颜色区分信号类型和采样周期，其中黑色表示连续信号，其他颜色表示离散信号。红色表示采样频率最高，绿色次之，黄色表示含多种采样频率。

5.4.3 间断点及代数环仿真建模

1. 间断点仿真建模

间断点在 MATLAB 中称为“系统过零”。Simulink 采用变步长求解器检测间断点。具体实现的方法是：对支持过零检测的模块，仿真执行时，Simulink 环境通知求解器缩小步长，直到满足误差要求为止。这种“通知”称为通知事件，每个支持过零检测的模块都有专属于自己的通知事件。

能够产生过零事件的库元素有：abs(数学库的求绝对值模块)、backlash(非线性库的偏移模块)、dead zone(非线性库的死区模块)、hit crossing(信号与系统库的零交叉模块)、integrator(连续库的积分模块)、minmax(数学库的最大、最小值模块)、relational operator(数学库的关系运算符模块)、relay(非线性库的延迟模块)、saturation(非线性库的饱和模块)、sign(数学库的符号运算模块)、step(输入库的阶跃模块)、subsystem(子系统库的子系统模块)、switch(非线性库的开关模块)等。在仿真这些库元素时，Simulink 默认使用过零

检测功能，如果用户不需要这种检测，可以在 simulation parameters 参数设置对话框中的 advanced 选项卡中关闭这项功能。

对不具有过零检测能力的库元素，如果需要进行过零检测，可以使用 signals & systems 中的 hit crossing 来实现，即当 hit crossing 的输入穿过某一偏移值(offset)时会产生一个过零事件。

模块过零产生的作用有两种类型：一是通知求解器，系统运行模式即动态特性是否发生了改变，二是驱动与之相关的其他模块。Simulink 中过零信号可分为三种类型：上升沿、下降沿、双边沿。

上升沿：信号上升到 0 或过 0，或由 0 变为正；

下降沿：信号下降到 0 或过 0，或由 0 变为负；

双边沿：任何信号的上升或下降沿的发生。

过零检测是需要花费时间的，因此对非真正间断点的过零检测通常是不必要的，可关闭过零事件检测以提高仿真速度。

2. 代数环仿真建模

代数环是指在所建立的用结构图描述的系统中，存在一种环路，在该环路上所有模块的输入与输出只存在代数关系。在系统的仿真结构图中，可能存在一些环节无法安排计算顺序。

例如图 5.3 表示的就是一个代数环，该回路只有一个求和模块，该模块的输出 z 也是它的输入。该代数环的数学表达式为

$$z = u - z$$

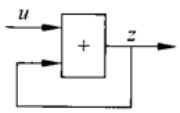


图 5.3 代数环

MATLAB 建模造成代数环的原因是某些库元素具有允许输入信号直通的能力，即输出直接依赖于输入，并且形成了回路。这些库元素包括：math function (数学函数模块)、gain(增益模块)、product(乘法模块)、state-space(状态空间模块，其中 D 不为 0)、transfer fcn(传递函数模块，分子与分母多项式阶次相同)、sum(求和模块)、zero-pole(零极点模块，零点与极点数目相同)，以及 integrator(积分模块)等。

对这种代数环模型，尽管 Simulink 采用 Newton-Raphson 方法解算，但难以保证收敛，因此应该避免系统模型中出现代数环。

如果实在避免不了代数环，那么应该事先对模型进行人工处理。如果系统的仿真结构图较复杂，且计算步长较小，可在回路中增加记忆模块(memory)、代数约束模块(algebraic constraint)等。如果系统的模型不复杂，可以将模型重写为一个标准的一阶微分方程组，方程右边不出现变量的导数，画出的仿真结构图就不会有代数环。

5.5 MATLAB 模块化集成技术

MATLAB 模块化集成技术，包括数据驱动模型、虚子系统与原子子系统技术等。

5.5.1 数据驱动模型

Simulink 采用模块化技术，基于数据驱动模型的思想，实现了建模、调试、仿真和分析

集成于一体。即模型的结构可以模型参数分离、模型与试验框架分离，模型参数及试验参数均可在模型结构确定后交互式定义。

Simulink 采用结构化建模实现系统模型的定义。基于模型图元库及交互式、图形化建模环境，建模时只需使用鼠标拖放库中的图元置于建模窗口中的适当位置，并按结构图的连接方式将其连接起来即可。Simulink 还具有分层多级建模能力，即可将某一层的某一模块进一步定义为多个模块的组合，并最终形成一个系统。Simulink 建模窗口可以交互式调整，以方便用户操作。

交互式仿真运行环境提供对某一系统的各个模块的参数独立定义的能力，这样，对同一结构的模型，用户可定义不同的参数而得到不同的实例，例如设计控制系统时，为比较不同控制系统参数对同一受控对象的影响，用户无须重新建立模型，只需修改参数。

交互式仿真运行环境提供对同一模型实例设置不同的仿真运行参数，如仿真运行的起始时间、结束时间、仿真方法、仿真步长、误差控制、输出分析要求设置等。

5.5.2 子系统集成

子系统最基本的应用是将一组相关的模块封装到单一模块之中，以便用户建模和仿真分析。从实现角度看，Simulink 子系统集成有虚子系统和原子子系统两种基本模式。

虚子系统，是指 Simulink 建模时，定义了一个虚模块，该模块在系统中的位置被定义（包括输入、输出），但其功能可能是多个模块的组合。除了通用子系统外，像条件执行子系统（conditional_subsystem）、使能子系统（enable_subsystem）均是虚子系统。对虚子系统，系统仿真执行时，子系统模块被转换到上一级框图中，并进行统一排序，模型执行的顺序与子系统本身无关，在一个仿真步长内，在虚子系统中定义的模块与上一级框图中的模块一样被执行，可多次进出同一个虚子系统。

原子子系统（atomic subsystem）的实现与虚子系统有很大不同。首先原子子系统虽然在上一级框图中被定位，并按顺序执行，但在一个仿真步长内，必须作为一个整体加以执行，而不能多次进出，相当于一个独立的系统。在原子子系统内，各模块仍按顺序执行。Simulink 定义原子子系统有两种方法，其一是在 subsystem 模块库中的 atomic subsystem 环境下定义一个空原子子系统，然后编辑建立；其二是将已经建立的子系统强制转换为原子子系统。需要注意的是，Simulink 规定，定义为使能子系统（enable_subsystem）的子系统，不能将其强制转换为原子子系统。

无论是虚子系统还是原子子系统，在系统建模时，必须逐一设置各模块的参数，为了更方便地使用，Simulink 提供了子系统的封装能力。“封装”就是将该子系统变成与 Simulink 中内置模块具有相同能力的子系统模块，即该模块一方面具有子系统的功能，又能像编辑内置模块一样自定义图标、设置参数、获得输入输出关系、拥有自己的工作区等。

5.5.3 MATLAB 典型子系统

1. 使能子系统（enabled subsystem）

使能子系统除输入和输出外，还有一个唯一的控制信号输入端口，被称为激活端口，只

有当它的控制信号输入为正时，也就是说当子系统被激活时，使能子系统才开始执行，并且只要控制信号保持为正它就一直执行。使能子系统的控制信号可以是标量或向量。如果控制信号是标量，当信号值大于零时子系统就执行；如果控制信号是向量，则当输入向量的任一元素的值大于零时，子系统就执行。

2. 触发子系统(triggered subsystem)

触发子系统除输入和输出外，也有一个唯一的控制信号输入端口，被称为触发端口，它决定子系统是否执行。触发子系统编辑窗口的 trigger 模块对话框中有四种触发事件，从中选择一种，强制一个触发子系统开始执行。

(1) 上升沿触发(rising)——在控制信号出现上升沿时开始执行；

(2) 下降沿触发(falling)——在控制信号出现下降沿时开始执行；

(3) 边沿触发(either)——在控制信号出现任何过零点时开始执行；

(4) 函数调用触发(function call)——执行与否取决于一个 S 函数的内部逻辑。

与使能子系统不同，触发子系统都具有零阶保持的特性。零阶保持，是指输出结果在触发事件之间保持不变。对于触发子系统而言，系统在触发信号控制下开始执行的时刻，系统由输入产生相应的输出；当触发信号产生过零点时，系统输出保持在原来的输出值，并不发生变化。此外，触发子系统的触发依赖于触发控制信号。因此对于触发子系统不能指定常值采样时间(即固定的采样时间)，只有带继承采样时间的模块才能够在触发子系统中应用。在这个系统之中，对于上升沿和下降沿触发的子系统，其采样周期为两个触发时刻之间的时间间隔(即触发控制方波信号的两个相邻上升沿或下降沿之间的间隔)；对于双边沿触发的子系统，其采样周期为触发控制方波信号相邻的上升沿或下降沿之间的间隔。

3. 原子子系统(atomic subsystem)

子系统都可以将系统中相关模块组合封装成一个单独的模块，大大方便了用户对复杂系统的建模、仿真和分析；但是对于不同的子系统而言，它们除了有如上的共同点之外，还存在着本质的不同。众所周知，无论对于通用子系统还是使能子系统，Simulink 在进行系统仿真时，子系统中的各个模块的执行并不受子系统的限制。也就是说，系统的执行与通用子系统或使能子系统的存在与否无关。这两种子系统的使用均是为了使 Simulink 框图模型形成一种层次结构，以增强系统模型的可读性。子系统中的模块在执行过程中与上一级的系统模块统一被排序，模块的执行顺序与子系统本身无关，在一个仿真时间步长之内，系统的执行可以多次进出同一个子系统。子系统相当于一种虚设的模块组容器，其中的各模块与系统中其他模块(子系统)的信号输入输出不受任何影响。因此，通用子系统与使能子系统这两种子系统，称为“虚子系统”。

触发子系统的工作原理与上述两种子系统的概念不同。在触发子系统中，当触发事件发生时，触发子系统中所有模块一同被执行。只有当子系统中的所有模块都被执行完毕后，Simulink 才会转移到系统模型中的上一层执行其他的模块。这与上述子系统中各模块的执行方式根本不同。这样的子系统称为“原子子系统”。

原子子系统与触发子系统在触发事件发生时，子系统中所有模块一同被执行；当子系

统中的所有模块都被执行完毕后，Simulink 才开始执行上一级其他模块或其他子系统。因此触发子系统为原子子系统。但是在有些情况下，需要将一个普通的子系统作为一个整体来执行，而不管它是不是触发子系统。这对于多速率复杂系统尤其重要，因为在多速率系统中，时序关系的任何差错，都会导致整个系统设计仿真的失败，而且难以进行诊断分析，尤其是在要生成可执行代码时更为重要。

在 Simulink 中有两种方法可以建立原子子系统：第一种，建立一个空的原子子系统。选择 subsystems 模块库中的 atomic subsystem 子系统模块，然后编辑原子子系统。第二种，将已经建立好的子系统强制转换为原子子系统。首先选择子系统，然后选择 Simulink 模型编辑器的 edit 菜单下的 block parameters 模块参数，框选 treat as atomic unit(作为原子子系统)即可；或单击鼠标右键，在弹出菜单中选择相同选项也可。注意：对于使能子系统而言，不能将其转换为原子子系统，这是因为使能子系统中模块执行顺序不能被改变。

4. 使能与触发子系统(enabled and triggered subsystem)

对于某些条件执行子系统而言，其控制信号可能不止一个。在很多情况下，条件执行子系统同时具有触发控制信号与使能控制信号，这样的条件执行子系统一般称为使能与触发子系统。顾名思义，使能与触发子系统指的是子系统的执行受到激活信号与触发信号的共同控制。也就是说，只有当激活条件与触发条件均满足的情况下，子系统才开始执行。使能与触发子系统的工作原理如下：系统等待一个触发事件的发生(也就是触发信号的产生)，当触发事件发生后，Simulink 检测使能控制信号，如果激活信号为正，则子系统执行一次，否则不执行子系统。由此可知，只有激活条件与触发条件均满足时，子系统才能够被执行。对于实际的动态系统而言，其中某些子系统很可能受到多个控制信号的控制(也就是系统输出受到多个条件的约束)，因此在相应的系统模型建立时，应该使用多个控制信号输入。用户在建立这样的条件执行子系统时，需要注意的是，在一个系统模块中不允许有多个 enable 信号或 trigger 信号。如果必须使用多个控制信号，用户可以使用逻辑操作符，先将相关的控制信号(即子系统执行条件)相组合，以产生单一的触发控制信号或使能控制信号。

5. 函数调用子系统(function call subsystem)

使用 S 函数的状态而非普通的信号作为触发子系统的控制信号。函数调用子系统属于触发子系统，在触发子系统的触发模块 trigger 的参数设置中选择 function-call，可以将普通信号触发的触发子系统转换为函数调用子系统。

6. While 循环子系统(while iterator subsystem)

与 For 循环子系统相类似，While 循环子系统同样可以在一个仿真时间步长之内循环执行子系统，但是其执行必须满足一定的条件。While 循环子系统有两种类型：当型与直到型，这与其他高级语言中的 While 循环类似。

7. 选择执行子系统(switch case action subsystem)

某些情况下，系统对于输入的不同取值，分别执行不同的功能。选择执行子系统必须同时使用 switch case 模块与 switch case action subsystem 模块(均在 subsystem 模块库中)。

8. For 循环子系统(for iterator subsystem)

For 循环子系统的目的是在一个仿真时间步长之内循环执行子系统。用户可以指定在一个仿真时间步长之内子系统执行的次数,以达到某种特殊的目的。

9. 表达式执行子系统(if action subsystem)

为了与前面的条件执行子系统相区别,这里我们称 if action subsystem 为表达式执行子系统。此子系统的执行依赖于逻辑表达式的取值,这与 C 语言中的 if else 语句类似。需要注意的是,表达式执行子系统必须同时使用 if 模块与 if action subsystem 模块(均在 subsystem 模块库中)。

10. 可配置执行子系统(configurable subsystem)

用来代表用户自定义库中的任意模块,只能在用户自定义库中使用。

5.6 小结

经过半个多世纪的发展,连续系统仿真语言已趋于成熟,能满足多种仿真的需要,并且能稳定可靠地运行。

本章以 MATLAB 为例介绍了连续系统仿真语言,重点介绍了动态仿真集成环境——Simulink 的功能特点和使用方法,Simulink 实际上就是根据前面几章介绍的仿真原理而编写的一种具有可视化界面的仿真软件包。与前面用微分方程和差分方程建模的传统仿真软件包相比,Simulink 具有更灵活、更方便的优点。

连续系统仿真语言的进一步发展更有效地支持分布式仿真、并行仿真的需要,以及适应连续/离散事件混合系统仿真的需要。

习题

1. 一个连续-离散时间混合系统,离散采样周期分别为 0.2s、0.3s、0.5s,试问如何选择仿真步长?

2. Simulink 有哪些方法可用于病态系统仿真?

3. MATLAB 是如何实现数据驱动模型的?

4. Simulink 是如何处理过零问题的?

5. Simulink 的虚子系统与原子子系统在模型定义及执行时有何区别?

6. 使用不同模型描述形式,在 MATLAB 中定义连续系统 $\frac{Y(s)}{U(s)}=\frac{s-1}{s^2+3s+4}$。

7. 在 Simulink 环境下建立如图 5.4 所示的控制系统模型,其中 $G(s)=\frac{25}{s(s+0.8)}$,$u(t)=1$,

$K_a=2$，$K_b=1$，采用定步长求解器进行仿真。

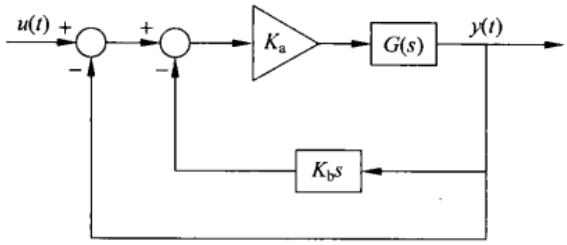


图 5.4 习题 7 控制系统模型

参考文献

[1] Brennan R D and Linebarger R N. A Survey of Digital Simulation. Simulation, Dec. 1964
[2] SCi Software Committee. The SCi Continuous-System Simulation Language. Simulation, Dec. 1967
[3] 肖田元等. 系统仿真导论. 北京：清华大学出版社，2000
[4] 熊光楞等. 连续系统仿真与离散事件系统仿真. 北京：清华大学出版社，1991
[5] 李兴玮，邱晓刚等. 计算机仿真技术基础. 长沙：国防科技大学出版社，2006
[6] 张志涌等. 精通 MATLAB 6.5 版. 北京：北京航空航天大学出版社，2003
[7] 姚俊，马松辉. Simulink 建模与仿真. 西安：西安电子科技大学出版社，2002
[8] 张若清，罗学科，王民. 控制工程基础及 MATLAB 实践. 北京：高等教育出版社，2008
[9] 刘卫国. MATLAB 程序设计与应用. 北京：高等教育出版社，2006
[10] 宋志安，徐瑞银. MATLAB 程序设计与应用. 北京：高等教育出版社，2008
[11] Cellier F E and Kofman E. Continuous system simulation. Springer, 2006
[12] 李国勇，谢克明，杨丽娟. 计算机仿真技术与 CAD. 北京：电子工业出版社，2008

第6章
分布参数系统仿真建模*

前面几章已就集中参数系统的仿真技术进行了讨论。实际上，真实的物理系统往往是分布参数系统，只是在研究时，为了突出系统的主要特征而将其进行必要的简化后才得到理论上的集中参数系统。但是，对某些实际的物理系统来说，不能或难于简化，或是只有按分布参数来考虑才能反映其运动规律。因此，本章讨论分布参数系统的仿真建模方法学。

6.1 概述

分布参数系统的运动规律一般采用偏微分方程(partial differential equation，PDE)来描述。对于确定型的偏微分方程，采用一阶描述形式，可以用以下表达式

$$F_0(\phi,p,z,t)\frac{\partial\phi}{\partial t}+\sum_{i=1}^{k}F_i(\phi,p,z,t)\frac{\partial\phi}{\partial z_i}=f(\phi,p,u,z,t) \tag{6.1}$$

$$\begin{cases} b(\phi,p,z,t)=0 \quad z\in\delta_z & (6.2)\\ \phi(z,0)=\phi^0(z) & (6.3)\\ y(z,t)=g(\phi,p,z,t) & (6.4)\\ h(\phi,p,u,z,t)\geqslant 0 & (6.5) \end{cases}$$

现对上述表达式中的有关符号说明如下：

(1) 自变量：常微分方程中自变量只有时间变量 t，而在偏微分方程中，其自变量除了时间 $t\in T$ 以外，还有空间自变量 $z\in Z$；

(2) 输入变量 $u\in U$ 及输入段集合：映射 $Z\times T\rightarrow U$：$u(z,t)$；

(3) 因变量：$\phi\in\Phi$，且是 z 与 t 的函数；

(4) 式(6.2)确定边界条件，δ_z 表示 Z 的边界，在 $z\in\delta_z$ 上，ϕ 随时间变化满足该等式；

(5) 式(6.3)表示初始条件，即规定初始时刻 ϕ 在域内的值；

(6) 输出变量 $y\in Y$ 是空间和时间的函数；

(7) 式(6.5)规定了约束条件。

当然,在某些情况下,系统是以高阶偏微分方程的形式给出的。一般来说,经过适当变换,高阶偏微分方程可以转换成一阶偏微分方程组。

偏微分方程的典型形式如下:

(1) 双曲方程,典型的有

对流方程

$$\frac{\partial u}{\partial t}+a\frac{\partial u}{\partial x}=q \tag{6.6}$$

波动方程

$$\frac{\partial^2 u}{\partial t^2}-a^2\frac{\partial^2 u}{\partial x^2}=q \tag{6.7}$$

(2) 抛物方程,典型的有

扩散方程

$$\frac{\partial u}{\partial t}-b\frac{\partial^2 u}{\partial x^2}=q \tag{6.8}$$

对流 - 扩散方程

$$\frac{\partial u}{\partial t}+a\frac{\partial u}{\partial x}-b\frac{\partial^2 u}{\partial x^2}=q \tag{6.9}$$

(3) 椭圆方程,典型的有泊松方程

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=q \tag{6.10}$$

从偏微分方程理论可以知道,为求得偏微分方程的数值解,初始条件的设置是十分重要的,即方程定义必须是充分完整的,既不能"过定"(过多的初始边界条件),也不能"欠定"(过少的边界条件),而只能"适定"。

例如,式(6.6)的一阶对流方程,假设 $a>0$,现要求在 $0<x<1$ 区间上的定解,在 $t=0$ 给定初值 $u(x,0)$,并给出 $x=0$ 的边值 $u(0,t)$,此种情况下 u 在该区间上的解可以唯一确定为 $u(x,t)=\varphi(x-at)$,且沿特征线 $x-at=\text{cons}\ t$ 为常值。我们说,此时,方程的初值是"适定"的。反之,如果此时还定义了 $x=1$ 的边值 $u(1,t)$,则是"过定"的;另一方面,如果只给定初值 $u(x,0)$,而未给出 $x=0$ 的边值 $u(0,t)$,则是"欠定"的;无论是"过定"还是"欠定",都无法确定方程的解。

又如,对于扩散方程式(6.8),含有二阶微商,应在左右两个边界上各给一个边界条件,而不能一边给两个条件,另一边一个条件也不给,势必导致边值扰动的不稳定性,这样的初值定义都是不恰当的。

6.2 差分法原理

经典的偏微分方程仿真建模的基本思路是:在空间和时间两个方面将系统离散化,从而将偏微分方程化为差分方程,然后从初值或边值出发,逐层推进,这种方法称为差分法。它具有高度的通用性,易于程序实现,有些文献也称为差分格式。为了说明差分法的原理,

先考虑空间为一维的情况。这样，偏微分方程的自变量为两个，即时间变量 t 和空间变量 x。

将 $x-t$ 平面的上半部($t\geqslant 0$)，用坐标线划分网格(参见图 6.1)

$$x = x_j = jh, \quad j = 0, \pm 1, \pm 2, \cdots$$
$$t = t_n = n\tau, \quad n = 0, \pm 1, \pm 2, \cdots$$

其中 h,τ 分别是空间和时间步长。

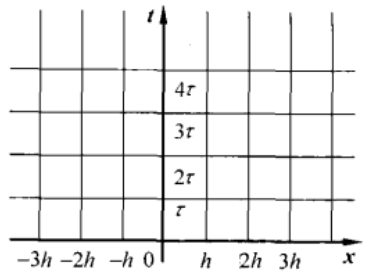


图 6.1 差分法原理图示

6.2.1 对流方程差分法

下面以对流方程为例来说明差分法的一些基本概念。对于式(6.6)的对流方程，已知初始条件为

$$u(x,t_0) = \varphi(x), \quad u(x_j,t_0) = \varphi(jh)$$
$$u(x_0,t) = \eta(t), \quad u(x_0,t_n) = \eta(n\tau)$$

简记为 $u_j^n = u(x_j,t_n)$，则用差分代替原方程的微分，可得到

$$\left.\frac{\partial u}{\partial t}\right|_{\substack{x=jh\\ t=n\tau}} = \frac{u_j^{n+1}-u_j^n}{\tau} \qquad \left.\frac{\partial u}{\partial x}\right|_{\substack{x=jh\\ t=n\tau}} = \frac{u_{j+1}^n - u_{j-1}^n}{\tau}$$

这里，对 x 的偏微分采用所谓的“中心差”公式(参见图 6.2)。这样，原偏微分方程成为如下的差分方程

$$\frac{1}{\tau}(u_j^{n+1}-u_j^n) + \frac{a}{2h}(u_{j+1}^n - u_{j-1}^n) - q_j^n = 0$$



图 6.2 一阶对流方程显式中心差公式

略加整理，可得到

$$u_j^{n+1} = u_j^n - \frac{a\tau}{2h}(u_{j+1}^n - u_{j-1}^n) + q_j^n \tag{6.11}$$

这称为一阶对流方程显式中心差公式。它可从初始条件 $u(x_j,t_0)=\varphi(jh)$，$u(x_0,t_n)=\eta(n\tau)$ 出发，沿时间轴一层一层地往前推进。下一层($n+1$ 层)用到的是前一层(n 层)的值，因而也叫做两层格式。以后将会看到，某些格式可能要用到前三层或多层的值，相应地称为三层格式或多层格式。

一阶对流方程的差分公式还有隐式形式，属两层格式，其表达式为

$$u_j^{n+1} = u_j^n - \frac{a\tau}{2h}(u_{j+1}^{n+1} - u_{j-1}^{n+1}) + q_j^{n+1} \tag{6.12}$$

式(6.12)是基于

$$\left.\frac{\partial u}{\partial t}\right|_{\substack{x=jh\\ t=n\tau}} = \frac{u_j^{n+1}-u_j^n}{\tau} \qquad \left.\frac{\partial u}{\partial x}\right|_{\substack{x=jh\\ t=n\tau}} = \frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{2h}$$

显然，在计算 u_j^{n+1} 时用到了 u_{j+1}^{n+1}，但此时后者尚未计算出来，因而称为“隐式”(参见图 6.3)。以后将会看到，隐式公式虽然实现起来比较困难，但具有稳定性好的优点。

按类似的做法，也可推导出扩散方程的差分模型。先介绍显式差分模型。由式(6.8)，用如下差分近似方程中的微分(参见图 6.4)

$$\left.\frac{\partial u}{\partial t}\right|_{\substack{x=jh\\t=n\tau}}=\frac{u_j^{n+1}-u_j^n}{\tau}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{\substack{x=jh\\t=n\tau}}=\left(\frac{u_{j+1}^n-u_j^n}{h}-\frac{u_j^n-u_{j-1}^n}{h}\right)\Big/h$$

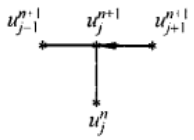


图6.3 一阶对流方程隐式中心差公式



图6.4 扩散方程显式公式

其中对 x 的二阶偏微分用二阶差分近似。这样，原微分方程的显式差分模型如下

$$\frac{u_j^{n+1}-u_j^n}{\tau}-b\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{h^2}-q_j^n=0 \tag{6.13}$$

略加整理，可得

$$u_j^{n+1}=\frac{b\tau}{h^2}u_{j+1}^n+\left(1-\frac{2b\tau}{h^2}\right)u_j^n+\frac{b\tau}{h^2}u_{j-1}^n+q_j^n \tag{6.14}$$

设 x 的区间为$[x_0,x_l]$，考虑初始条件

$$u(x,t_0)=\varphi(x),u(x_0,t)=\xi_1(t),u(x_l,t)=\xi_2(t)$$

则从 $t=t_0$ 时的 $u_j^0=u(x_j,t_0)=\varphi(jh)(j=1,2,\cdots,M-1)$，及 $x=x_0$ 与 $x=x_l$ 时的 $u_0^n=u(x_0,t_m)=\xi_1(n\tau)(n=1,2,\cdots,N)$；$u_l^n=u(x_l,t_m)=\xi_2(n\tau)$出发(其中在$(t_0,x_0)$时 $u_0^0=\varphi(0)=\xi_1(0)$，在$(t_0,x_l)$时 $u_l^0=\xi_2(0)$)，按时间推进，逐层得到

$$u_j^{n+1}\quad(j=1,2,\cdots,M;\ n=1,2,\cdots,N-1)$$

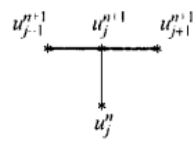


图6.5 扩散方程隐式公式

按同样思想，若(参见图6.5)

$$\left.\frac{\partial u}{\partial t}\right|_{\substack{x=jh\\t=n\tau}}=\frac{u_j^{n+1}-u_j^n}{\tau}$$

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{\substack{x=jh\\t=n\tau}}=\left(\frac{u_{j+1}^{n+1}-u_j^{n+1}}{h}-\frac{u_j^{n+1}-u_{j-1}^{n+1}}{h}\right)\Big/h$$

这是扩散方程的隐式差分公式。将其代入原微分方程式(6.8)，略加整理，可得到

$$u_j^{n+1}=\frac{1}{1-\frac{b\tau}{h^2}}\left[u_j^n+\frac{b\tau}{h^2}(u_{j+1}^{n+1}+u_{j-1}^{n+1})+\tau q_j^{n+1}\right] \tag{6.15}$$

上述扩散方程的显式及隐式也都是两层格式。

6.2.2 差分格式的相容性、收敛性及稳定性

当用差分方程近似微分方程时，自然会带来误差。为了描述这种误差的大小，仿真分析和数值分析一样，采用截断误差的概念。

以扩散方程为例，设 $u(x,t)$ 是式(6.8)的充分光滑解，若将其在(x_j,t_n)处进行泰勒展开，可得到

$$\frac{u(x_j,t_{n+1})-u(x_j,t_n)}{\tau}-b\frac{u(x_{j+1},t_n)-2u(x_j,t_n)+u(x_{j-1},t_n)}{h^2}-q_j^n$$

$$=\left[\frac{\partial u}{\partial t}-b\frac{\partial^2 u}{\partial x^2}\right]_j^n-q_j^n+\frac{1}{2}\tau\left[\frac{\partial^2 u}{\partial t^2}\right]_j^{n+\theta_1}-\frac{b}{24}h^2\left(\left[\frac{\partial^4 u}{\partial x^4}\right]_{j+\theta_2}^n+\left[\frac{\partial^4 u}{\partial x^4}\right]_{j+\theta_3}^n\right)=0$$

其中 $0\leqslant(\theta_1,\theta_2,\theta_3)\leqslant 1$。这样，式(6.13)中

$$u(x_j,t_{n+1})=u(x_j,t_n)+\tau\left[\frac{\partial u}{\partial t}\right]_j^n+\frac{1}{2}\tau^2\left[\frac{\partial^2 u}{\partial t^2}\right]_j^{n+\theta_1}$$

$$u(x_{j+1},t_n)=u(x_j,t_n)+h\left[\frac{\partial u}{\partial x}\right]_j^n+\frac{1}{2}h^2\left[\frac{\partial^2 u}{\partial x^2}\right]_j^n+\frac{1}{6}h^3\left[\frac{\partial^3 u}{\partial x^3}\right]_j^n+\frac{1}{24}h^4\left[\frac{\partial^4 u}{\partial x^4}\right]_{j+\theta_2}^n$$

$$u(x_{j-1},t_n)=u(x_j,t_n)-h\left[\frac{\partial u}{\partial x}\right]_j^n+\frac{1}{2}h^2\left[\frac{\partial^2 u}{\partial x^2}\right]_j^n-\frac{1}{6}h^3\left[\frac{\partial^3 u}{\partial x^3}\right]_j^n+\frac{1}{24}h^4\left[\frac{\partial^4 u}{\partial x^4}\right]_{j+\theta_3}^n$$

考虑原方程的解是充分光滑的，所以上述各式的最后一项都是有界的。将上述各式代入式(6.13)后，再减去式(6.13)，当 $\tau\to 0,h\to 0$ 时，有

$$E=\frac{1}{2}\tau\left[\frac{\partial^2 u}{\partial t^2}\right]_j^{n+\theta_1}-\frac{b}{24}h^2\left(\left[\frac{\partial^4 u}{\partial x^4}\right]_{j+\theta_2}^n+\left[\frac{\partial^4 u}{\partial x^4}\right]_{j+\theta_3}^n\right)=O(\tau)+O(h^2)$$

则称 E 是差分格式式(6.13)的截断误差。有时把 $O(\tau)+O(h^2)$ 简记为 $O(\tau+h^2)$。如果一个差分格式的截断误差是 $O(\tau^q+h^p)$，那么，就称该差分格式对时间 t 是 q 阶精度，对空间是 p 阶精度。

用一个差分方程近似一个偏微分方程时，如果当时间步长 τ 和空间步长 h 趋于 0 时，这个差分方程的截断误差也趋于 0，则称该差分方程与原偏微分方程是相容的，或称该差分方程具有**相容性**。相容性是差分法中的一个基本概念。一般说来，用一个差分模型来近似微分模型，相容性条件必须满足。

下面分析差分格式的稳定性。如前所述，利用差分方程对偏微分方程仿真时，仿真计算是按时间层逐层推进的，而每一层都会有计算误差(包括舍入误差和截断误差)。我们当然希望误差始终保持在一定的范围，而不至于越来越大，这就是所谓的**稳定性**问题。

以扩散方程的显式差分公式为例，该公式具有相容性，其截断误差为 $O(\tau+h^2)$。设初值 u_j^0 有一误差 ε_j^0，即 $(u+\varepsilon)_j^0=u_j^0+\varepsilon_j^0$，在差分格式下，$u_j^n$ 的仿真结果为 $(u+\varepsilon)_j^n=u_j^n+\varepsilon_j^n$，代入差分方程式(6.13)，得到

$$\frac{1}{\tau}[(u+\varepsilon)_j^{n+1}-(u+\varepsilon)_j^n]-\frac{b}{h^2}[(u+\varepsilon)_{j-1}^n-2(u+\varepsilon)_j^n+(u+\varepsilon)_{j+1}^n]-q_j^n=0 \tag{6.16}$$

将式(6.16)与式(6.13)相减，可得

$$\frac{1}{\tau}(\varepsilon_j^{n+1}-\varepsilon_j^n)-\frac{b}{h^2}(\varepsilon_{j-1}^n-2\varepsilon_j^n+\varepsilon_{j+1}^n)=0 \tag{6.17}$$

用谐波分析法求误差差分方程的解，设 ε_j^0 的简谐波的形式表示为

$$\varepsilon_j^0=(\mathrm{e}^{\mathrm{j}kx})_{x=jh}=\mathrm{e}^{\mathrm{j}kjh}$$

其中 k 为频率参数，也称波数。定义

$$\varepsilon(jh,n\tau)=\mathrm{e}^{n\mu\tau}(k)(\mathrm{e}^{\mathrm{j}kx})\Big|_{\substack{x=jh\\ t=n\tau}}=(\lambda(k))^n\mathrm{e}^{\mathrm{j}kjh}\quad(j=0,\pm1,\pm2,\cdots;\ n=0,1,\cdots)$$

其中 $\lambda(k)=\mathrm{e}^{\mu\tau}(k)$ 为对应波数 k 的增长因子，将其代入式(6.17)，简记 $\lambda(k)$ 为 λ，经整理后得到

$$\frac{1}{\tau}(\lambda-1)-\frac{b}{h^2}(e^{-jkh}-2+e^{jkh})=0 \tag{6.18}$$

称为误差方程式(6.17)的特征方程，其特征根为

$$\lambda=1-\frac{2\tau b}{h^2}(1-\cos kh)=1-4\beta\sin^2\frac{\theta}{2} \tag{6.19}$$

其中 $\beta=2\tau b/h^2$，$\theta=kh$(以下同)。显然，只有当特征根的绝对值不大于1时，误差方程才能保证误差不随时间加大而增长，从而该扩散方程的显式差分格式才是稳定的，即其稳定的条件是：$\beta\leqslant 1/2$，即 $\tau\leqslant h^2/2b$，也就是说，式(6.14)是条件稳定的。

对于一个适定的初值问题，如果逼近它的差分格式是和它相容的，且具有稳定性，则该差分模型收敛于原偏微分方程，称该差分格式具有**收敛性**。初值问题是线性的，它具有这种性质，对于非线性问题则可能没有这样简单的结论。

按照同样的做法，可以得到扩散方程隐式差分格式以及其他偏微分方程的差分模型的稳定条件。表6.1及表6.2中分别给出了典型的双曲方程及抛线方程的差分模型的形式、相应的截断误差及其稳定条件。

表6.1 典型的双曲方程差分模型及其稳定条件

$$\left\{\begin{array}{l}\dfrac{\partial u}{\partial t}+a\dfrac{\partial u}{\partial x}-q=0\\ \alpha=a\tau/h,\theta=kh\end{array}\right\}$$

名称	格式与截差	特征方程与稳定条件
右偏显式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{h}(u_{j+1}^n-u_j^n)-q_j^n=0$ $E=O(\tau+h)$	$\lambda=1+\alpha-\alpha e^{j\theta}$ $-1\leqslant\alpha\leqslant 0$
左偏显式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{h}(u_j^n-u_{j-1}^n)-q_j^n=0$ $E=O(\tau+h)$	$\lambda=1-\alpha+\alpha e^{j\theta}$ $0\leqslant\alpha\leqslant 1$
右偏隐式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{h}(u_{j+1}^{n+1}-u_j^{n+1})-q_j^{n+1}=0$ $E=O(\tau+h)$	$\lambda=(1-\alpha+\alpha e^{j\theta})^{-1}$ $\alpha\leqslant 0$ 或 $1\leqslant\alpha$
左偏隐式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{h}(u_j^{n+1}-u_{j-1}^{n+1})-q_j^{n+1}=0$ $E=O(\tau+h)$	$\lambda=(1+\alpha-\alpha e^{j\theta})^{-1}$ $\alpha\leqslant -1$ 或 $0\leqslant\alpha$
中心差显式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)-q_j^n=0$ $E=O(\tau+h^2)$	$\lambda=1-j\alpha\sin\theta$ 恒不稳
中心差隐式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^{n+1}-u_{j-1}^{n+1})-q_j^{n+1}=0$ $E=O(\tau+h^2)$	$\lambda=(1+j\alpha\sin\theta)^{-1}$ 恒稳
耗散中心差	$\frac{1}{\tau}\left[u_j^{n+1}-\frac{1}{2}(u_{j+1}^n+u_{j-1}^n)\right]$ $+\frac{a}{2h}(u_{j-1}^n-2u_j^n+u_{j+1}^n)-q_j^n=0$ $E=O(\tau^2/h)+O(\tau+h)$	$\lambda=\cos\theta+j\alpha\sin\theta$ $\lvert\alpha\rvert\leqslant 1$
菱形	$\frac{1}{2\tau}(u_j^{n+2}-u_j^n)+\frac{a}{2h}(u_{j+1}^{n+1}-u_{j-1}^{n+1})-q_j^{n+1}=0$ $E=O(\tau^2+h^2)$	$\lambda_{1,2}=j\alpha\sin\theta\pm\sqrt{1-\alpha^2\sin^2\theta}$ $\alpha\leqslant 1\quad\lvert\lambda\rvert=1$

续表

名称	格式与截差	特征方程与稳定条件
跳点	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)-q_j^n=0$ $(n+1+j=\text{偶})$ $\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^{n+1}-u_{j-1}^{n+1})-q_j^{n+1}=0$ $(n+1+j=\text{奇})$ $E=O(\tau^2+h^2)$	$\lambda_{1,2}=\mathrm{j}\alpha\sin\theta\pm\sqrt{1-\alpha^2\sin^2\theta}$ $\alpha\leqslant 1 \quad \lvert\lambda\rvert=1$

$$\begin{bmatrix}\frac{\partial^2 u}{\partial t^2}-a^2\frac{\partial^2 u}{\partial x^2}-q=0,a>0\\ \delta_t^2 u_j^{n+1}=u_j^{n+2}-2u_j^{n+1}+u_j^n\\ \delta_x^2 u_j^n=u_{j-1}^n-2u_j^n+u_{j+1}^n\\ \alpha=a\tau/h,\theta=kh\end{bmatrix}$$

名称	格式与截差	特征方程与稳定条件
显式	$\frac{1}{\tau^2}\delta_t^2 u_j^{n+1}-\frac{a^2}{h^2}\delta_x^2 u_j^n-q_j^{n+1}=0$ $E=O(\tau^2+h^2)$	$\lambda^2+\left(4\alpha^2\sin^2\frac{\theta}{2}-2\right)\lambda+1=0$ $\alpha^2\leqslant 1 \quad \tau\leqslant h/a$
隐式	$\frac{1}{\tau^2}\delta_t^2 u_j^{n+1}-\frac{a^2}{h^2}\delta_x^2 u_j^{n+2}-q_j^{n+1}=0$ $E=O(\tau+h^2)$	$\left(1+4\alpha^2\sin^2\frac{\theta}{2}\right)\lambda^2+2\lambda+1=0$ 恒稳
平均隐式	$\frac{1}{\tau^2}\delta_t^2 u_j^{n+1}-\frac{1}{2}\left[\frac{a^2}{h^2}\delta_x^2 u_j^n+\frac{a^2}{h^2}\delta_x^2 u_j^{n+2}\right]-q_j^{n+1}=0$ $E=O(\tau^2+h^2)$	$\left(1+2\alpha^2\sin^2\frac{\theta}{2}\right)\lambda^2-2\lambda$ $+\left(1+2\alpha^2\sin^2\frac{\theta}{2}\right)=0$ 恒稳

表 6.2 典型的抛物方程差分模型及其稳定条件

$$\begin{bmatrix}\frac{\partial u}{\partial t}-b\frac{\partial^2 u}{\partial x^2}-q=0\\ \delta^2 u_j^n=u_{j-1}^n-2u_j^n+u_{j+1}^n,\beta=\tau b/h^2,\theta=kh\end{bmatrix}$$

名称	格式与截差	特征方程与稳定条件
显式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)-\frac{b}{h^2}\delta^2 u_j^n-q_j^n=0$ $E=O(\tau+h^2)$	$\lambda=1-4\beta\sin^2\frac{\theta}{2}$ $\beta\leqslant\frac{1}{2}$
隐式	$\frac{1}{2\tau}(u_j^{n+1}-u_j^n)-\frac{b}{h^2}\delta^2 u_j^{n+1}-q_j^{n+1}=0$ $E=O(\tau+h^2)$	$\lambda=\left(1+4\beta\sin^2\frac{\theta}{2}\right)^{-1}$ 恒稳
中心差	$\frac{1}{2\tau}(u_j^{n+2}-u_j^n)-\frac{b}{h^2}\delta^2 u_j^{n+1}-q_j^{n+1}=0$ $E=O(\tau^2+h^2)$	$\lambda^2+(4\beta-4\beta\cos\theta)\lambda+1=0$ $\lambda_{1,2}=-4\sin^2\frac{\theta}{2}\pm\sqrt{1+\left(4\beta\sin^2\frac{\theta}{2}\right)^2}$ 恒不稳

续表

名称	格式与截差	特征方程与稳定条件
菱形	$\frac{1}{2\tau}(u_j^{n+2}-u_j^n)-\frac{b}{h^2}(u_{j-1}^{n+1}-u_j^{n+2}-u_j^n+u_{j+1}^{n+1})-q_j^{n+1}=0$ $E=O(\tau^2/h^2)+O(\tau^2+h^2)$	$(1+2\beta)\lambda^2-(4\beta\cos\theta)\lambda-(1-2\beta)=0$ $\lambda_{1,2}=(2\beta\cos\theta \pm \sqrt{1-4\beta^2\sin^2\theta})(1+2\beta)$ 恒稳
跳点	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)-\frac{b}{h^2}\delta^2 u_j^n-q_j^n=0$ $(n+1+j=\text{偶})$ $\frac{1}{\tau}(u_j^{n+1}-u_j^n)-\frac{b}{h^2}\delta^2 u_j^{n+1}-q_j^{n+1}=0$ $(n+1+j=\text{奇})$ $E=O(\tau^2/h^2)+O(\tau^2+h^2)$	$\lambda_{1,2}=(2\beta\cos\theta \pm \sqrt{1-4\beta^2\sin^2\theta})(1+2\beta)$ 恒稳

$$\begin{cases}\dfrac{\partial u}{\partial t}+a\dfrac{\partial u}{\partial x}-a^2\dfrac{\partial^2 u}{\partial x^2}-q=0, a\leqslant 0, b\geqslant 0\\ \delta^2 u_j^n=u_{j-1}^n-2u_j^n+u_{j+1}^n\\ \alpha=a\tau/h, \beta=b\tau/h^2, \theta=kh\end{cases}$$

名称	格式与截差	特征方程与稳定条件
中心差显式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)$ $-\frac{b}{h^2}\delta^2 u_j^n-q_j^n=0$ $E=O(\tau^2+h^2)$	$\lambda=1-2\beta+2\beta\cos\theta-\mathrm{j}\alpha\sin\theta$ $2\beta\leqslant 1 \quad \frac{\alpha^2}{2\beta}\leqslant 1$
中心差全隐式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^{n+1}-u_{j-1}^{n+1})$ $-\frac{b}{h^2}\delta^2 u_j^{n+1}-q_j^{n+1}=0$ $E=O(\tau+h^2)$	$\lambda=(1+2\beta-2\beta\cos\theta+\mathrm{j}\alpha\sin\theta)^{-1}$ 恒稳
菱形	$\frac{1}{2\tau}(u_j^{n+2}-u_j^n)+\frac{a}{2h}(u_{j+1}^{n+1}-u_{j-1}^{n+1})$ $-\frac{b}{h^2}(u_{j-1}^{n+1}-u_j^{n+2}-u_j^n+u_{j+1}^{n+1})-q_j^{n+1}=0$ $E=O(\tau^2/h^2)+O(\tau^2+h^2)$	$(1+2\beta)\lambda^2-(2\beta\cos\theta-\mathrm{j}\alpha\sin\theta)\lambda-(1-2\beta)=0$ $\lvert\alpha\rvert\leqslant 1 \quad \tau\leqslant h/\lvert\alpha\rvert$
跳点	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)$ $-\frac{b}{h^2}\delta^2 u_j^n-q_j^{n+1}=0,\ (n+1+j=\text{偶})$ $\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^{n+1}-u_{j-1}^{n+1})$ $-\frac{b}{h^2}\delta^2 u_j^{n+1}-q_j^{n+1}=0,\ (n+1+j=\text{奇})$ $E=O(\tau^2/h^2)+O(\tau^2+h^2)$	$(1+2\beta)\lambda^2-(2\beta\cos\theta-\mathrm{j}\alpha\sin\theta)\lambda-(1-2\beta)=0$ $\lvert\alpha\rvert\leqslant 1 \quad \tau\leqslant h/\lvert\alpha\rvert$

6.3 典型分布参数系统的差分法

6.3.1 扩散方程的菱形法及跳点法

从表 6.1 及表 6.2 中可以看到，双曲方程和抛物方程有多种不同形式的差分法，从而可得到不同形式的仿真模型。前面已经讨论了两层格式的基本方法，本节以扩散方程的菱形法和跳点法为例讨论初值问题的多层格式。

扩散方程式(6.8)的显式公式及隐式公式的截断误差对时间是一阶精度。从仿真计算的角度来看，似乎是太低了，人们希望能提高截断误差的阶次，因而提出了如下中心差公式

$$\frac{1}{2\tau}(u_j^{n+2}-u_j^n)-\frac{b}{h^2}(u_{j-1}^{n+1}-2u_j^{n+1}+u_{j+1}^{n+1})-q_j^{n+1}=0 \tag{6.20}$$

从表 6.2 中可以看到，该差分格式为恒不稳，无法直接使用，尽管其截断误差为 $E=O(\tau^2+h^2)$。

如果令 $u_j^{n+1}=\frac{1}{2}(u_j^{n+2}+u_j^n)$，将其代入式(6.20)，可得

$$\frac{1}{2\tau}(u_j^{n+2}-u_j^n)-\frac{b}{h^2}(u_{j-1}^{n+1}-u_j^{n+2}-u_j^n+u_{j+1}^{n+1})-q_j^{n+1}=0$$

略加整理，可得到

$$\left(1+\frac{2\tau b}{h^2}\right)u_j^{n+2}=\left(1-\frac{2\tau b}{h^2}\right)u_j^n+\frac{2\tau b}{h^2}(u_{j-1}^{n+1}+u_{j+1}^{n+1})+2\tau q_j^{n+1} \tag{6.21}$$

即为求 u_j^{n+2}，仅用到 u_j^n，u_{j-1}^{n+1}，u_{j+1}^{n+1} 三点的值，这四点构成一个菱形，因而称为菱形法(参见图 6.6)。

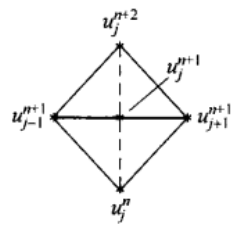


图 6.6 中心差及菱形公式

菱形法的稳定性如何呢？观察其特征方程

$$(1+2\beta)\lambda^2-(4\beta\cos\theta)\lambda-(1-2\beta)=0$$

其中 $\beta=\tau b/h^2$，$\theta=kh$。不难得到其特征根为

$$\lambda_{1,2}=(2\beta\cos\theta\pm\sqrt{1-4\beta^2\sin^2\theta})/(1+2\beta)$$

显然，$|\lambda_{1,2}|\leqslant 1$，这意味着菱形法为恒稳，而且它是一种显式方法，截断误差 $E=O(\tau^2/h^2)+O(\tau^2+h^2)$ 为时间的二阶精度。

菱形法虽然具有很好的稳定性，但是却不具有相容性。下面来讨论这一问题。设 $u(x,t)$ 是扩散方程式(6.8)的光滑解，在 (x_j,t_n) 处进行泰勒展开，可得其截断误差为

$$E=b\left(\frac{\tau}{h}\right)^2\left[\frac{\partial^2 u}{\partial t^2}\right]_j^n+O(\tau^2+h^2)+O\left(\frac{\tau^4}{h^2}\right)$$

可见，为保证式(6.21)与原扩散方程的相容性，要求当 $\tau\to 0$，$h\to 0$ 时，τ/h 必须趋向于 0。这就是说，菱形法满足相容性的充分必要条件是：τ 趋向于 0 的速度必须大于 h 趋向于 0 的速度。反之，如果 τ/h 保持不变，则式(6.21)就不具有与原扩散方程的相容性。此时，与带阻尼的波动方程相容。显然差分方程式(6.20)的解将不收敛于原扩散方程 $\frac{\partial u}{\partial t}-b\frac{\partial^2 u}{\partial x^2}+$

$\gamma^2 b\dfrac{\partial^2 u}{\partial t^2}-q=0$。

由这个例子可以看到，在用差分法建立偏微分方程的模型时，不仅要考虑方法的稳定性，而且要验证方法的相容性，这一点与常微分方程是有区别的。在常微分方程中，一种方法只要满足稳定性，就必然满足相容性，从而就具有收敛性。

为了使菱形法满足相容性，可以选 $\tau=O(h^{1+s})(0<s<1)$，此时截断误差 $E=O(\tau^2+h^2)+O(h^{2s})$，精度比中心差的精度略低。

菱形法是三层法，这给实际使用带来两方面的困难：(1)需要保存前两层的数据，内存开销大；(2)对初值问题，无法自启动，必须用其他方法来启动它。为此，人们对菱形法作了进一步的改进。

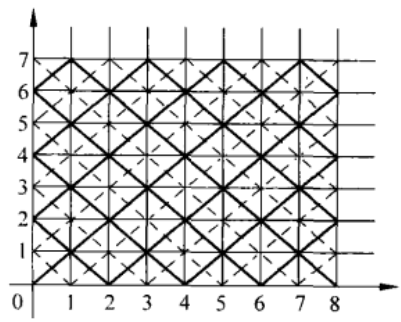


图 6.7　跳点格式的示意图

设想将时空节点(j,n)按照 $n+j=$奇数(虚线通过的点)或偶数(实线通过的点)分为两组，从而形成两套互相交错且互相独立的菱形格网(参见图 6.7)。

若从 n 层推进到 $n+1$ 层，先在偶点用显式公式

$$\frac{1}{\tau}(u_j^{n+1}-u_j^n)-\frac{b}{h^2}(u_{j-1}^n-2u_j^n+u_{j+1}^n)-q_j^n=0\quad(n+j=\text{偶})$$

产生新值，然后在奇点用全隐式公式

$$\frac{1}{\tau}(u_j^{n+1}-u_j^n)-\frac{b}{h^2}(u_{j-1}^{n+1}-2u_j^{n+1}+u_{j+1}^{n+1})-q_j^{n+1}=0\quad(n+j=\text{奇})$$

计算。这时 u_j^{n+1} 的左右 u_{j-1}^{n+1}，u_{j+1}^{n+1} 均为新值，因此，表面上是隐式，实际上是显式。

实际上，上述用隐式计算得到的奇点 u_j^{n+1} 不必保存，因为下一层的偶点 u_j^{n+2} 计算式

$$\frac{1}{\tau}(u_j^{n+2}-u_j^{n+1})-\frac{b}{h^2}(u_{j-1}^{n+1}-2u_j^{n+1}+u_{j+1}^{n+1})-q_j^{n+1}=0\quad(n+2+j=\text{偶})$$

将该式与奇点的隐式相减，即

$$u_j^{n+2}=2u_j^{n+1}-u_j^n\quad(n+j+2=\text{偶})$$

也就是说，采用上述隐式计算得到 u_j^{n+1} 后，直接进行上式计算得到 u_j^{n+2}，所有的偶点也不必用显式计算。这就是所谓的“跳点法”。

综上所述，跳点法用于扩散方程仿真时，在初始层 $u_j^0(j=1,2,\cdots)$ 及 $u_0^n(n=0,1,2,\cdots)$ 的基础上，首先对 $n=1$ 层的偶点使用一次显式产生新值，以后按隐式计算奇点，不保存，直接计算出下一层的偶点的值加以保存，一直进行下去，直到最后一层才补齐奇点的值。

实际上，将 $u_j^{n+2}=2u_j^{n+1}-u_j^n\rightarrow u_j^{n+1}=\dfrac{1}{2}(u_j^{n+2}+u_j^n)$ 代入隐式后，消去 u_j^{n+1} 即得到菱形公式，不过只在偶点网格上进行计算，跳过了奇点，跳点法由此而得名。这意味着，跳点公式具有菱形法的恒稳和显式特性，克服了菱形法三层格式不能自启动和需要大存储量的缺点，并节约了大约一半工作量。当然，也保留了菱形法的相容性及计算精度方面的缺点。

6.3.2 对流方程的耗散中心差公式

对于对流方程

$$\frac{\partial u}{\partial t}+a\frac{\partial u}{\partial x}-q=0$$

显式中心差公式

$$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)-q_j^n=0 \tag{6.22}$$

该公式为恒不稳，难以直接使用。若令(参见图 6.8)

$$u_j^n=\frac{1}{2}(u_{j-1}^n+u_{j+1}^n) \tag{6.23}$$

则显式中心差公式成为如下形式

$$\frac{1}{\tau}\left(u_j^{n+1}-\frac{1}{2}(u_{j-1}^n+u_{j+1}^n)\right)+\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)-q_j^n=0$$

略加整理，可得

$$u_j^{n+1}=-\frac{a\tau-h}{2h}u_{j+1}^n+\frac{a\tau+h}{2h}u_{j-1}^n-\tau q_j^n$$

可以得到其特征方程为

$$\lambda-\cos(kh)-\mathrm{j}\frac{a}{h}\sin(kh)=0$$

令 $\theta=kh$，$\alpha=a/h$，则有

$$\lambda=\cos\theta-\mathrm{j}\alpha\sin\theta \tag{6.24}$$

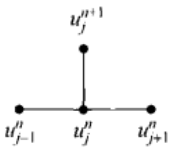


图 6.8 耗散中心差

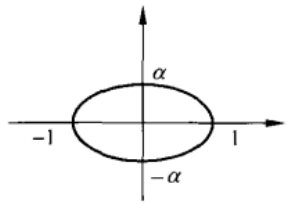


图 6.9 耗散中心差特征根轨迹

特征根 λ 的轨迹是以原点为中心，横半径为 1，纵半径为 $|\alpha|$ 的椭圆(参见图 6.9)，稳定条件为

$$|\alpha|\leqslant 1,\quad \tau\leqslant h/|a| \tag{6.25}$$

亦称为库朗条件。实际上，根据式(6.23)式(6.22)可表示为

$$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)-\frac{1}{2\tau}(u_{j-1}^n-2u_j^n+u_{j+1}^n)-q_j^n=0$$

其中第三项

$$-\frac{1}{2\tau}(u_{j-1}^n-2u_j^n+u_{j+1}^n)\approx-\frac{h^2}{2\tau}\frac{\partial^2 u}{\partial x^2} \tag{6.26}$$

称为耗散项，从而该差分公式称为耗散中心差公式。

将式(6.26)在节点(x_j,t_n)作幂次展开，并与原微分方程相减，可得其截断误差

$$E = -\left(\frac{h^2}{2\tau}\right)\left(\frac{\partial^2 u}{\partial x^2}\right) + O(\tau + h) \tag{6.27}$$

当 $h,\tau \to 0$ 并取 $\tau = O(h)$，则 $E = O(h) + O(\tau + h) = O(h)$，即具有一阶精度。但是，如果 $h,\tau \to 0$ 时 $h^2/\tau = \sigma$ 保持为常值(例如取 $\tau = O(h^2)$)，则由式(6.27)可知差分方程式(6.26)的极限形式就不再是原对流方程，而是对流-扩散方程

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} - \frac{\sigma}{2}\frac{\partial^2 u}{\partial x^2} - q = 0 \quad (\sigma = h^2/\tau) \tag{6.28}$$

多出了一个扩散项。因此，在计算时，时间步长 τ 相对于 h 不能取得太小，否则因耗散太大而造成的误差太大，无法得到可信的仿真结果。

6.3.3 椭圆方程的差分法

物理上的定常态问题，如静电磁场、引力场，热传导(温度分布)及扩散(浓度分布)等，通常可以用椭圆型微分方程来描述。典型的有两种形式

(1) 拉普拉斯方程

$$-\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0 \tag{6.29}$$

(2) 泊松方程

$$-\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = f \tag{6.30}$$

这类问题大多是在某种边界条件下，求解椭圆方程以得到 $u = u(x,y)$，因而亦称为边值问题。这里仅讨论边界条件以 $u(x,y) = a(x,y)$形式给出时的仿真计算问题。

下面以泊松方程为例，讨论如何用差分法建立其仿真模型。设 x,y 为区域 Ω 上的坐标变量，将其分别以 h,k 为间隔进行等分

$$x = x_i = ih \quad (i = 0,1,2,\cdots,M)$$
$$y = y_j = jk \quad (j = 0,1,2,\cdots,N)$$

则 $u_{ij} = u(ih,jk)$。

先不考虑 Ω 的不规则性，即认为 Ω 恰好可被 h,k 分别在 x,y 方向等分，其节点 u_{ij} 可分为两类：内点($i=1,2,\cdots,M-1$；$j=1,2,\cdots,N-1$)和边点($\{i=0;\ j=0,1,2,\cdots,N\}$，$\{i=M;\ j=0,1,2,\cdots,N\}$，$\{j=0;\ i=1,2,\cdots,M-1\}$，$\{j=N;\ i=1,2,\cdots,M-1\}$)。采用差分法，对内节点

$$\begin{cases} \left.\dfrac{\partial^2 u}{\partial x^2}\right|_{i,j} = \dfrac{1}{h^2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) + O(h^2) \\ \left.\dfrac{\partial^2 u}{\partial y^2}\right|_{i,j} = \dfrac{1}{k^2}(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}) + O(k^2) \end{cases}$$

代入原方程，得到差分方程为

$$2\left(\frac{1}{h^2} + \frac{1}{k^2}\right)u_{i,j} - \frac{1}{h^2}(u_{i-1,j} + u_{i+1,j}) - \frac{1}{k^2}(u_{i,j-1} + u_{i,j+1}) = f_{i,j}$$
$$(i = 1,2,\cdots,M-1;\ j = 1,2,\cdots,N-1) \tag{6.31}$$

方程中包括五个节点，因而称为五点差分格式，特别是，当取 $h = k$ 时，式(6.31)变得非常

简洁

$$4u_{i,j}-(u_{i-1,j}+u_{i+1,j}+u_{i,j-1}+u_{i,j+1})=h^2f_{i,j} \tag{6.32}$$

若 $M=N$,则得到$(N-1)(N-1)$个差分方程和四组边界条件值如下

$$\begin{cases}4u_{i,j}-(u_{i-1,j}+u_{i+1,j}+u_{i,j-1}+u_{i,j+1})=h^2f_{i,j} \quad (i,j=1,2,\cdots,N-1)\\ u_{i,j}=a_{i,j} \quad (i=0,N;\ j=0,1,2,\cdots,N) \text{ 或} (j=0,N;\ i=1,2,\cdots,N-1)\end{cases} \tag{6.33}$$

求解上述方程的基本方法是迭代法。在迭代过程中,边点的值保持不变,内点的迭代表达式如下

$$u_{i,j}^{m+1}=\frac{1}{4}(h^2f_{i,j}+u_{i-1,j}^{m}+u_{i+1,j}^{m}+u_{i,j-1}^{m}+u_{i,j+1}^{m})$$

$$(i,j=1,2,\cdots,N-1) \tag{6.34}$$

这是一种简单的迭代法,可以证明,其迭代过程是收敛的,即无论 $u_{i,j}$ 的初值 $u_{i,j}^{(0)}$ 怎样取,当 $m\to\infty$ 时,$u_{i,j}^{(m)}$ 恒收敛于式(6.33)的真解。

简单迭代法的缺点是:(1)收敛速度慢,(2)每点产生的新值不能立即冲掉原有值,因为后者在计算其邻节点时还需要用到,这样,内存开销就很大。由于这些原因,这种方法在实际中很少使用。可以将上述方法加以改进。Seidel 提出一种新的迭代法,称为松弛法。其基本思想是:在每点新值计算出来时,立即刷新原有值,下一相邻点计算时采用该点的新值,为此,要求在计算时规定一个扫描顺序。

例如,若先按行号 $i=1,2,\cdots$,每行内按列号 $j=1,2,\cdots$。对节点(i,j),其邻点$(i-1,j)$,$(i,j-1)$均为新值,从而松弛法的计算公式为

$$u_{i,j}^{m+1}=\frac{1}{4}(h^2f_{i,j}+u_{i-1,j}^{m+1}+u_{i+1,j}^{m}+u_{i,j-1}^{m+1}+u_{i,j+1}^{m})$$

$$(i,j=1,2,\cdots,N-1) \tag{6.35}$$

将其改写成增量形式:

$$u_{i,j}^{m+1}=u_{i,j}^{m}+\frac{1}{4}(h^2f_{i,j}+u_{i-1,j}^{m+1}+u_{i+1,j}^{m}+u_{i,j-1}^{m+1}+u_{i,j+1}^{m}-4u_{i,j}^{m})$$

$$(i,j=1,2,\cdots,N-1) \tag{6.36}$$

进一步,引入松弛因子 ω,式(6.36)成为

$$u_{i,j}^{m+1}=u_{i,j}^{m}+\frac{\omega}{4}(h^2f_{i,j}+u_{i-1,j}^{m+1}+u_{i+1,j}^{m}+u_{i,j-1}^{m+1}+u_{i,j+1}^{m}-4u_{i,j}^{m})$$

$$(i,j=1,2,\cdots,N-1) \tag{6.37}$$

可以证明,当松弛因子取值在 $0<\omega<2$ 区间时,式(6.37)是收敛的,而且松弛因子若取最优值时,收敛速度大大加快。当 $\omega>1$ 时称为超松弛,最优松弛因子取值在 $1<\omega^*<2$ 之间,属于超松弛。

需要说明的是,简单迭代法每次迭代的结果不依赖于扫描顺序,松弛法则不同,即不同的扫描顺序,即使最优松弛因子取相同的值,初期的有限次数内收敛速度也是不相同的。幸运的是,最终的收敛速度是相同的。

以上讨论了当区域 Ω 具有规则的边界时的迭代法。实际问题当然往往不是这样。区域边界不规则,内存的安排和扫描控制将变得复杂些。一种简单的方法是将边界不规则的网格套在一个规则的网格之内(参见图 6.10),并把节点分为两类:一类是原有节点,即计算

点，另一类是“虚节点”，即非需要的计算点。迭代扫描时按规则网格进行，对“虚节点”是空跑，这是通过增加存储空间和计算时间的开销来换取程序控制的简化。

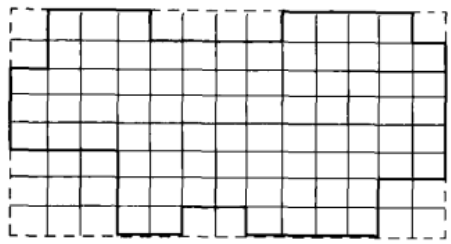


图 6.10 不规则边界的处理

6.4 线上求解法

到目前为止，我们讨论的各种方法均是基于经典的差分法，它是完全独立于常微分方程的各种数值解法发展起来的。

考虑到常微分方程已有大量成熟的算法，20 世纪 70 年代以来，基于常微分方程仿真方法发展起来的另一大类偏微分方程仿真建模方法——线上求解法（method on lines）得到广泛应用。

实际上，偏微分方程与常微分方程的区别在于，前者有一个或多个空间变量作为自变量。线上求解法的基本思想是：将空间变量进行离散化，而时间变量仍保持连续，从而将偏微分方程转化为一组常微分方程，进而基于常微分方程的各种仿真算法进行仿真。下面就几类典型偏微分方程的线上求解方法进行原理性说明。

（1）对流方程

考察一般形式的对流方程

$$\frac{\partial u}{\partial t}+a\frac{\partial u}{\partial x}-q=0$$

将 x 所在的区域划分若干个子区间，得 $x_i=ih(i=0,1,2,\cdots,M)$，这样，在 x_i 处，原偏微分方程变成常微分方程

$$\left.\frac{\mathrm{d}u}{\mathrm{d}t}\right|_{x_i}=-a\left(\frac{\partial u}{\partial x}\right)_{x_i,t}+q \tag{6.38}$$

显然，在 t_0 处，$\partial u/\partial x|x_i,t=t_0$ 的值可以由原方程的初值求得（比如说采用差分法），从而，式(6.38)成为

$$\left.\frac{\mathrm{d}u}{\mathrm{d}t}\right|_{x_i}=f_i(u,t) \tag{6.39}$$

这样，由 $M+1$ 个常微分方程组成的方程组从初值出发，按照统一的步长往前推进，就可以得到整个模型的解。

在常微分方程的算法保证计算的稳定性及计算精度的前提下，线上求解法的最主要的问题是如何对$\partial u/\partial x|x_i,t=t_k$ 进行求值。从前面各节的讨论中已经知道，各种差分法（无论是显式还是隐式）均具有一定的精度和稳定域，从而整个模型的仿真精度和稳定性直接受这

些方法的影响。

三次样条插值具有较高的精度，下面介绍基于三次样条插值计算$\partial u/\partial x|_{x_i,t=t_k}$的原理。

在初始时刻$t=t_0$，x各点$x_i(i=0,1,2,\cdots,M)$的u_i值是已知的。对每个子区间$[x_i, x_{i+1}]$，三次多项式样条函数$p_i(x)$可由左右两端点处的二阶导数值L_i，R_i来构造

$$p_i(x)=\frac{L_i}{6h_i}(x_{i+1}-x)^3+\frac{R_i}{6h_i}(x-x_i)^3+\left(\frac{u_{i+1}}{h_i}-\frac{h_iR_i}{6}\right)(x-x_i)+\left(\frac{u_i}{h_i}-\frac{h_iL_i}{6}\right)(x_{i+1}-x) \tag{6.40}$$

若对$p_i(x)$进行微分，并分别令其下标为$i-1$，i，可得

$$\begin{cases} p_i'(x)=-\dfrac{1}{3}h_iL_i-\dfrac{1}{6}h_iR_i+\dfrac{u_{i+1}-u_i}{h_i} & (x=x_i) \\ p_{i-1}'(x)=-\dfrac{1}{3}h_{i-1}R_{i-1}-\dfrac{1}{6}h_{i-1}R_{i-1}+\dfrac{u_i-u_{i-1}}{h_{i-1}} & (x=x_{i-1}) \end{cases} \tag{6.41}$$

考虑样条函数一阶导数连续，$p_{i-1}'(x_i)=p_i'(x_i)$，并记

$$\Delta_i=\frac{u_{i+1}-u_i}{h_i},\Delta_{i-1}=\frac{u_i-u_{i-1}}{h_{i-1}}$$

则由式(6.41)可得

$$h_i(2L_i+R_i)+h_{i-1}(2R_i+L_i)=6(\Delta_i-\Delta_{i-1}) \tag{6.42}$$

考虑$R_{i-1}=L_i$，并记为ϕ_i，可得

$$h\phi_{i-1}+2(h_{i-1}+h_i)\phi_i+h_i\phi_{i+1}=6(\Delta_i-\Delta_{i-1}) \tag{6.43}$$

由初始条件可知Δ_{i-1}，Δ_i，可由子区间的端点值确定两个定解条件，即可得到$\phi_i(i=1,2,\cdots,M)$，从而可得$p_i(x)$，进而得到$p_i'(x)$，即求得$\partial u/\partial x|_{x_i,t}$。

(2) 扩散方程

典型的扩散方程如下式描述

$$\frac{\partial u}{\partial t}-b\frac{\partial^2 u}{\partial x^2}-q=0$$

仍将x分成若干个子区间，考虑最常见的情况$x_i=ih(i=0,1,2,\cdots,M)$，对某个x_i，不难得到

$$\left.\frac{\mathrm{d}u}{\mathrm{d}t}\right|_{x_i,t}=b\left(\frac{\partial^2 u}{\partial x^2}\right)_{x_i,t}+q \tag{6.44}$$

这样可得$M+1$个微分方程，其中$\partial^2 u/\partial x^2|_{x_i,t}$可以用二阶差分来近似，即

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{x_i,t}=f_i(u,t)=[u_{i+1}(t)-2u_i(t)+u_{i-1}(t)]/h^2 \tag{6.45}$$

(3) 双自变量二阶偏微分方程

二阶偏微分方程的一般形式为

$$a\frac{\partial^2 u}{\partial x^2}+b\frac{\partial^2 u}{\partial x\partial y}+c\frac{\partial^2 u}{\partial y^2}+d\frac{\partial u}{\partial x}+e\frac{\partial u}{\partial y}+f=0 \tag{6.46}$$

当$a=0$，$b=0$时，可将x视为时间变量

$$\frac{\partial u}{\partial x}=-\frac{c}{d}\frac{\partial^2 u}{\partial y^2}-\frac{e}{d}\frac{\partial u}{\partial y}-\frac{1}{d}f \tag{6.47}$$

仍可用线上求解法仿真。

对一般情况，则需要进行适当变换。令

$$\frac{\partial u}{\partial y}=\gamma \tag{6.48}$$

将 y 视为时间变量，则

$$\frac{\partial \gamma}{\partial y}=-\frac{b}{c}\frac{\partial \gamma}{\partial x}-\frac{e}{c}\gamma-\frac{1}{c}\left(a\frac{\partial^2 u}{\partial x^2}+d\frac{\partial u}{\partial x}+f\right) \tag{6.49}$$

初始条件为边界值 $u(x,y_0)=G(x)$，$u(x_0,y)=K(y)$，$\partial u/\partial y|_{y=y_0}=H_0(x)$。

仿真过程中要采用所谓“交替”线上求解法，其基本原理说明如下：

① 由 $y=y_0$ 及初值 $u(x,y_0)=G(x)$ 出发；

② 对 $x_i(i=0,1,2,\cdots)$，由 $\gamma_0=\partial u/\partial y|_{y=y_0}=H_0(x)$ 计算 $\gamma(x_i,y_0)$ 的值；

③ 沿 x_i 向前推进 δy 到 $y_1=y_0+\delta y$，按线上求解法求解式 $\dfrac{\partial u}{\partial y}=\gamma$，计算得到 $u(x_i,y_1)$；

④ 式(6.49)也推进 δy，也进行线上求解，计算得到 $\gamma(x_i,y_1)$。

重复③、④两步，从而可计算出任何位置的 u 值。

线上求解法的原理比较简单，它充分利用了常微分方程仿真算法的优点，仅在一个自变量方向采用差分法计算，既直观又易于实行。仿真过程中，数值积分与差分交替进行。在使用这种方法时，正确选择差分方法以实现对空间变量求导，这是保证仿真模型稳定性及计算精度的前提。

6.5 有限元方法预备知识

有限元方法(finite element method，FEM)这一术语首次出现于 Clough R W 在 1960 年 9 月发表的一篇讨论平面弹性力学问题的论文中。20 世纪 70 年代是有限元方法的蓬勃发展时期，今天，随着科学与工程的发展，特别是在高性能计算机支持下，有限元方法得到广泛的应用。

有限元方法较之采用直交网格的差分法具有独特的优越性，易于适应区域形状的任意性，也易于用计算机程序实现。有限元方法的内容十分丰富，本书仅做原理性介绍。需要深入学习和掌握该方法的读者建议进一步参考有关专门书籍。

6.5.1 微分方程的等效积分形式与加权余量法

对于一般形式的偏微分方程，如用以下形式表示

$$\boldsymbol{A}(\boldsymbol{u})=\begin{Bmatrix}A_1(\boldsymbol{u})\\A_2(\boldsymbol{u})\\\vdots\end{Bmatrix}=0 \quad (\text{在 }\Omega\text{ 内}) \tag{6.50}$$

其中 $\boldsymbol{u}$ 是未知变量，可以是标量场(如温度)，也可以是几个变量组成的向量场(如位移、应变、应力等)。域 Ω 可以是体积、面积域，同时未知变量 $\boldsymbol{u}$ 还应满足边界条件

$$\boldsymbol{B}(\boldsymbol{u})=\begin{Bmatrix}B_1(\boldsymbol{u})\\B_2(\boldsymbol{u})\\\vdots\end{Bmatrix}=0 \quad (\text{在 } \Gamma \text{ 内}) \tag{6.51}$$

Γ 是 Ω 的边界，$\boldsymbol{A}$、$\boldsymbol{B}$ 是表示对于独立变量(如时间、空间坐标)的微分算子。

由于微分方程式(6.50)在域 Ω 中每一点必须为 0，因此就有

$$\int_{\Omega}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{A}(\boldsymbol{u})\mathrm{d}\Omega\equiv\int_{\Omega}(v_1A_1(\boldsymbol{u})+v_2A_2(\boldsymbol{u})+\cdots)\mathrm{d}\Omega\equiv 0 \tag{6.52}$$

其中 $\boldsymbol{V}=\begin{Bmatrix}v_1\\v_2\\\vdots\end{Bmatrix}$ 是函数向量，它是一组与微分方程个数相等的任意函数。式(6.52)是与式(6.50)完全等效的积分形式，即可以断言，若式(6.52)对于任意 $\boldsymbol{V}$ 都能成立，则式(6.50)必然在域内任何一点均能满足。同样，如果满足边界条件式(6.51)，则有一组任意函数 $\bar{\boldsymbol{V}}$，下式应该成立

$$\int_{\Gamma}\bar{\boldsymbol{V}}^{\mathrm{T}}\boldsymbol{B}(\boldsymbol{u})\mathrm{d}\Gamma\equiv\int_{\Gamma}(\bar{v}_1B_1(\boldsymbol{u})+\bar{v}_2B_2(\boldsymbol{u})+\cdots)\mathrm{d}\Gamma\equiv 0 \tag{6.53}$$

因此，对于所有 $\boldsymbol{V}$ 和 $\bar{\boldsymbol{V}}$ 都成立的积分形式

$$\int_{\Omega}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{A}(\boldsymbol{u})\mathrm{d}\Omega+\int_{\Gamma}\bar{\boldsymbol{V}}^{\mathrm{T}}\boldsymbol{B}(\boldsymbol{u})\mathrm{d}\Gamma=0 \tag{6.54}$$

等效于满足式(6.50)与式(6.51)。称式(6.54)是微分方程的等效积分形式。当然 $\boldsymbol{V}$ 和 $\bar{\boldsymbol{V}}$ 的选取必须满足式(6.54)是可积分计算的。

对于复杂的实际问题，式(6.50)与式(6.51)的精确解是很难得到的，若采用近似函数来表示，一般形式是

$$\boldsymbol{u}\approx\bar{\boldsymbol{u}}=\sum_{i=1}^{n}N_ia_i=\boldsymbol{N}\boldsymbol{a} \tag{6.55}$$

即近似函数是一族带有待定参数的已知函数，其中 a_i 是待定参数，N_i 是已知函数，它取自完全的函数序列，是线性独立的。完全的函数系列是指任何函数都可以用此序列来表示。近似解通常满足强制边界条件和连续性要求。

通常 n 取有限项，显然，近似解不能精确满足原微分方程，必然会有残差 $\boldsymbol{R}$、$\bar{\boldsymbol{R}}$，称为余量。

$$\boldsymbol{A}(\boldsymbol{N}\boldsymbol{a})=\boldsymbol{R}\boldsymbol{B}(\boldsymbol{N}\boldsymbol{a})=\bar{\boldsymbol{R}} \tag{6.56}$$

对式(6.54)，如果用 n 个已知函数 $\boldsymbol{W}_j$，$\bar{\boldsymbol{W}}_j$ 分别代替任意函数 $\boldsymbol{V}_j$，$\bar{\boldsymbol{V}}_j$，则其近似积分形式如下

$$\int_{\Omega}\boldsymbol{W}_j^{\mathrm{T}}\boldsymbol{A}(\boldsymbol{N}\boldsymbol{a})\mathrm{d}\Omega+\int_{\Gamma}\bar{\boldsymbol{W}}_j^{\mathrm{T}}\boldsymbol{B}(\boldsymbol{N}\boldsymbol{a})\mathrm{d}\Gamma=0 \tag{6.57}$$

也可以写成余量形式

$$\int_{\Omega}\boldsymbol{W}_j^{\mathrm{T}}\boldsymbol{R}\mathrm{d}\Omega+\int_{\Gamma}\bar{\boldsymbol{W}}_j^{\mathrm{T}}\bar{\boldsymbol{R}}\mathrm{d}\Gamma=0 \tag{6.58}$$

式(6.57)与式(6.58)的意义是：通过选择待定系数 a_i，强迫余量在某种平均意义上等于 0，而 $\boldsymbol{W}_j$，$\bar{\boldsymbol{W}}_j$ 称为权函数。余量的加权积分为 0 就得到了一组微分方程，近似求解该微分方程组以得到待定系数 a_i，由于函数 $\boldsymbol{N}$ 为已知，从而得到原问题的近似解。这就是加权余量法。

求解式(6.57)的展开式是

$$\begin{aligned}
&\int_{\Omega} \boldsymbol{W}_1^{\mathrm{T}} \boldsymbol{A}(\boldsymbol{Na})\mathrm{d}\Omega + \int_{\Gamma} \overline{\boldsymbol{W}}_1^{\mathrm{T}} \boldsymbol{B}(\boldsymbol{Na})\mathrm{d}\Gamma = \boldsymbol{0}\\
&\int_{\Omega} \boldsymbol{W}_2^{\mathrm{T}} \boldsymbol{A}(\boldsymbol{Na})\mathrm{d}\Omega + \int_{\Gamma} \overline{\boldsymbol{W}}_2^{\mathrm{T}} \boldsymbol{B}(\boldsymbol{Na})\mathrm{d}\Gamma = \boldsymbol{0}\\
&\qquad\vdots\\
&\int_{\Omega} \boldsymbol{W}_n^{\mathrm{T}} \boldsymbol{A}(\boldsymbol{Na})\mathrm{d}\Omega + \int_{\Gamma} \overline{\boldsymbol{W}}_n^{\mathrm{T}} \boldsymbol{B}(\boldsymbol{Na})\mathrm{d}\Gamma = \boldsymbol{0}
\end{aligned}$$

若微分方程 $\boldsymbol{A}$ 的个数为 m_1，边界条件 $\boldsymbol{B}$ 的个数为 m_2，则权函数 $\boldsymbol{W}_j$，$\overline{\boldsymbol{W}}_j$ $(j=1,2,\cdots,n)$ 分别为 m_1 阶和 m_2 阶的函数列阵。

按照对权函数的选择，可分别得到不同的加权余量法，通常有以下几种方法：

① 配点法

$$\boldsymbol{W}_j = \overline{\boldsymbol{W}}_j = \delta(x - x_j) \tag{6.59}$$

若 Ω 是独立坐标 x 的函数，则 $\delta(x-x_j)$ 具有以下性质：当 $x \neq x_j$ 时 $W_j=0$，但有

$$\int_{\Omega} \boldsymbol{W}_j \mathrm{d}\Omega = \boldsymbol{I} \quad (j = 1,2,\cdots,n) \tag{6.60}$$

这相当于简单地强迫余量在域内 n 个点上等于 0。

② 子域法

在 n 个子域 Ω_j 内 $\boldsymbol{W}_j=\boldsymbol{I}$，在子域 Ω_j 以外 $\boldsymbol{W}_j=\boldsymbol{0}$，此方法实质上是强迫余量在 n 个子域 Ω_j 的积分为 0。

③ 最小二乘法

当近似解取为 $\tilde{\boldsymbol{u}} = \sum_{i=1}^{n} N_i a_i$ 时，权函数 $\boldsymbol{W}_j = \dfrac{\partial}{\partial a_i}\boldsymbol{A}\left(\sum_{i=1}^{n} N_i a_i\right)$，此方法的实质是使得函数 $\boldsymbol{I}(a_i) = \int_{\Omega} \boldsymbol{A}^2 \left(\sum_{i=1}^{n} N_i a_i\right)\mathrm{d}\Omega$ 取最小值，即 $\partial \boldsymbol{I}/\partial a_i = 0$，$(i=1,2,\cdots,n)$。

④ 力矩法

以一维为例，令 $W_j = 1, x, x^2, \cdots$，得到

$$\int_{\Omega} A(\tilde{u})\mathrm{d}x = 0, \quad \int_{\Omega} xA(\tilde{u})\mathrm{d}x = 0, \quad \int_{\Omega} x^2 A(\tilde{u})\mathrm{d}x = 0, \cdots \tag{6.61}$$

该方法是强迫余量的各阶矩等于 0，通常又称为积分法。

⑤ 伽辽金(Galerkin)法

用试探函数序列作为权函数，即取 $\boldsymbol{W}_j = \boldsymbol{N}_j$，在边界上 $\overline{\boldsymbol{W}}_j = -\boldsymbol{N}_j$，则式(6.57)可写成

$$\int_{\Omega} \boldsymbol{N}_j^{\mathrm{T}} \boldsymbol{A}\left(\sum_{i=1}^{n} N_i a_i\right)\mathrm{d}\Omega - \int_{\Gamma} \boldsymbol{N}_j^{\mathrm{T}} \boldsymbol{B}\left(\sum_{i=1}^{n} N_i a_i\right)\mathrm{d}\Gamma = 0 \quad (j = 1,2,\cdots,n) \tag{6.62}$$

在很多情况下，采用伽辽金法得到求解方程的系数矩阵是对称的，因而这种方法得到大多数人的青睐。

下面通过一个简单的例子说明加权余量法用不同权函数的解题过程和结果比较。

例 6.1 *求解二阶微分方程*

$$\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} + u + x = 0 \quad (0 \leqslant x \leqslant 1) \tag{①}$$

边界条件

$$\text{当 } x=0 \text{ 时},u=0;\ \text{当 } x=1 \text{ 时},u=0 \qquad ②$$

取近似解为

$$u=x(1-x)(a_1+a_2x+\cdots) \qquad ③$$

显然,近似解满足边界条件②,但不满足微分方程①,在域内产生余量 R。余量加权积分为 0 的表达式如下

$$\int_0^1 W_iR\,\mathrm{d}x=0 \qquad ④$$

近似解可取③中的一项、两项或更多项。显然,项数越多,精度越高。下面以两项为例说明加权余量法。近似解的表达式为

$$\tilde{u}=x(1-x)(a_1+a_2x) \qquad ⑤$$

代入①,可得余量为

$$R(x)=x+a_1(-2+x-x^2)+a_2(2-6x+x^2-x^3) \qquad ⑥$$

(1) 配点法　因为取两项,所以在域内选两个配点,不妨取三分点 $x=1/3$ 及 $x=2/3$,得到余量表达式为

$$R\left(\frac{1}{3}\right)=\frac{1}{3}-\frac{16}{9}a_1+\frac{2}{27}a_2=0 \quad R\left(\frac{2}{3}\right)=\frac{2}{3}-\frac{16}{9}a_1-\frac{50}{27}a_2=0$$

解得

$$a_1=0.194\,8,a_2=0.173\,1$$

从而,两项近似解为

$$\tilde{u}=x(1-x)(0.194\,8+0.173\,1x)$$

(2) 子域法　两项近似解的两个子域选择如下

$$\text{当 } 0\leqslant x\leqslant\frac{1}{2}(\Omega_1) \text{ 时 } W_1=1;\quad \text{当 } \frac{1}{2}<x\leqslant 1(\Omega_2) \text{ 时 } W_2=1$$

由④得到

$$\begin{aligned}\int_0^{1/2}R(x)\mathrm{d}x&=\int_0^{1/2}[x+a_1(-2+x-x^2)+a_2(2-6x+x^2-x^3)]\mathrm{d}x\\&=\frac{1}{8}-\frac{11}{12}a_1+\frac{53}{192}a_2=0\\\int_{1/2}^1R(x)\mathrm{d}x&=\int_{1/2}^1[x+a_1(-2+x-x^2)+a_2(2-6x+x^2-x^3)]\mathrm{d}x\\&=\frac{3}{8}-\frac{11}{12}a_1-\frac{229}{192}a_2=0\end{aligned}$$

解得

$$a_1=\frac{291}{1\,551}=0.187\,6,\quad a_2=\frac{24}{141}=0.170\,2$$

从而,两项近似解为

$$\tilde{u}=x(1-x)(0.187\,6+0.170\,2x)$$

(3) 最小二乘法　将余量的二次方 R^2 在域 Ω 中积分,将该积分对待定系数求导等于 0

$$I=\int_\Omega R^2\mathrm{d}\Omega,\quad \text{并且}\frac{\partial I}{\partial a_i}=\int_\Omega R\frac{\partial R}{\partial a_i}\mathrm{d}\Omega=0\quad(i=1,2)$$

可得

$$\int_0^1R\frac{\partial R}{\partial a_1}\mathrm{d}x=\int_0^1[x+a_1(-2+x-x^2)+a_2(2-6x+x^2-x^3)](-2+x-x^2)\mathrm{d}x=0$$

$$\int_0^1 R\frac{\partial R}{\partial a_2}\mathrm{d}x=\int_0^1[x+a_1(-2+x-x^2)+a_2(2-6x+x^2-x^3)](2-6x+x^2-x^3)\mathrm{d}x=0$$

联立解上述两个方程，可得

$$a_1=0.1875, a_2=0.1695$$

从而，两项近似解为

$$\tilde{u}=x(1-x)(0.1875+0.1695x)$$

(4) 力矩法　取 $W_1=1, W_2=x$，对余量加权积分

$$\int_0^1 R(x)\mathrm{d}x=\int_0^1[x+a_1(-2+x-x^2)+a_2(2-6x+x^2-x^3)]\mathrm{d}x=0$$

$$\int_0^1 xR(x)\mathrm{d}x=\int_0^1[x^2+a_1x(-2+x-x^2)+a_2x(2-6x+x^2-x^3)]\mathrm{d}x=0$$

解得

$$a_1=0.1880, a_2=0.1695$$

从而，两项近似解为

$$\tilde{u}=x(1-x)(0.1880+0.1695x)$$

(5) 伽辽金法　取近似函数作为权函数

因为

$$\tilde{u}=N_1a_1+N_2a_2=a_1x(1-x)+a_2x_2(1-x)$$

取权函数

$$W_1=N_1=x(1-x);\quad W_2=N_2=x^2(1-x)$$

对余量加权积分

$$\int_0^1 W_1R(x)\mathrm{d}x=\int_0^1 x(1-x)[x+a_1(-2+x-x^2)+a_2(2-6x+x^2-x^3)]\mathrm{d}x=0$$

$$\int_0^1 W_2R(x)\mathrm{d}x=\int_0^1 x^2(1-x)[x^2+a_1x(-2+x-x^2)+a_2x(2-6x+x^2-x^3)]\mathrm{d}x=0$$

解得

$$a_1=0.1924, a_2=0.1707$$

从而，两项近似解为

$$\tilde{u}=x(1-x)(0.1924+0.1707x)$$

该方程的精确解是 $u=\dfrac{\sin x}{\sin 1}-x$。读者可以验证，取两项的近似解所有方法的误差均小于2%，尤以伽辽金法精度最高，其他方法的精度基本差不多。 ■

6.5.2 弹性力学的基本方程

为了说明有限元法原理，我们以弹性力学为对象。为此，首先对弹性力学的基本方程加以说明。

弹性体在载荷作用下，体内任意一点的应力状态可由6个应力分量 $\sigma_x,\sigma_y,\sigma_z,\tau_{xy},\tau_{yz},\tau_{zx}$ 来表示。其中 $\sigma_x,\sigma_y,\sigma_z$ 为正应力，$\tau_{xy},\tau_{yz},\tau_{zx}$ 为剪应力。应力分量及其正方向见图6.11，其正方向是如下规定的：如果某一个平面的外法线方向与坐标轴的正方向一致，这个面上的应力分量就以沿坐标轴正方向为正，与坐标轴反方向为负。相反，如果某一个平面的外法线方向与

坐标轴的负方向一致，这个面上的应力分量就以沿坐标轴负方向为正，与坐标轴同方向为负。

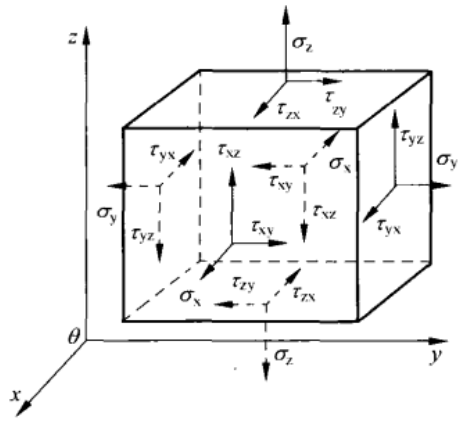


图 6.11 应力向量

应力分量的矩阵表示称为应力列阵或应力向量

$$\boldsymbol{\sigma}=\begin{Bmatrix}\sigma_x\\ \sigma_y\\ \sigma_z\\ \tau_{xy}\\ \tau_{yz}\\ \tau_{zx}\end{Bmatrix}=\begin{bmatrix}\sigma_x & \sigma_y & \sigma_z & \tau_{xy} & \tau_{yz} & \tau_{zx}\end{bmatrix}^{\mathrm{T}}$$

弹性体在载荷的作用下，还将产生位移和变形。弹性体内任一点的位移可由沿直角坐标轴方向的 3 个位移分量 u,v,w 来描述，可表示为

$$\boldsymbol{u}=\begin{Bmatrix}u\\ v\\ w\end{Bmatrix}=\begin{bmatrix}u & v & w\end{bmatrix}^{\mathrm{T}}$$

称为位移向量。弹性体内任一点的应变，可以由 6 个应变分量 $\varepsilon_x,\varepsilon_y,\varepsilon_z,\gamma_{xy},\gamma_{yz},\gamma_{zx}$ 来表示，其中 $\varepsilon_x,\varepsilon_y,\varepsilon_z$ 为正应变，$\gamma_{xy},\gamma_{yz},\gamma_{zx}$ 为剪应变。应变的正负号与应力的正负号对应，即应变以伸长为正，缩短为负，剪应变是以两个沿坐标轴正方向的线段组成的直角变小为正，反之为负，可表示为

$$\boldsymbol{\varepsilon}=\begin{bmatrix}\varepsilon_x,\varepsilon_y,\varepsilon_z,\gamma_{xy},\gamma_{yz},\gamma_{zx}\end{bmatrix}^{\mathrm{T}}$$

对于三维问题，弹性力学基本方程可写成如下形式：

① 平衡方程

弹性体 V 域内任一点沿坐标轴 x,y,z 方向的平衡方程为

$$\begin{cases}\dfrac{\partial\sigma_x}{\partial x}+\dfrac{\partial\tau_{yx}}{\partial y}+\dfrac{\partial\tau_{zx}}{\partial z}+\bar{f}_x=0\\ \dfrac{\partial\tau_{xy}}{\partial x}+\dfrac{\partial\sigma_y}{\partial y}+\dfrac{\partial\tau_{zy}}{\partial z}+\bar{f}_y=0\\ \dfrac{\partial\tau_{xz}}{\partial x}+\dfrac{\partial\tau_{yz}}{\partial y}+\dfrac{\partial\sigma_z}{\partial z}+\bar{f}_z=0\end{cases}\tag{6.63}$$

其中$\bar{f}_x,\bar{f}_y,\bar{f}_z$为单位体积的体积力在$x,y,z$方向的分量。写成矩阵形式为

$$\boldsymbol{A}\boldsymbol{\sigma}+\bar{\boldsymbol{f}}=\boldsymbol{0}(\text{在 } V \text{ 内}) \tag{6.64}$$

其中$\boldsymbol{A}$为微分算子

$$\boldsymbol{A}=\begin{bmatrix}\partial/\partial x & 0 & 0 & \partial/\partial y & 0 & \partial/\partial z\\ 0 & \partial/\partial y & 0 & \partial/\partial x & \partial/\partial z & 0\\ 0 & 0 & \partial/\partial z & 0 & \partial/\partial y & \partial/\partial x\end{bmatrix} \tag{6.65}$$

$\bar{\boldsymbol{f}}$是体积力向量，$\bar{\boldsymbol{f}}^{\mathrm{T}}=[\bar{f}_x \quad \bar{f}_y \quad \bar{f}_z]$。

② 几何方程——应变-位移关系

在微小位移与微小变形的情况下，略去位移导数的高次幂，则应变向量和位移向量间的几何关系可表示为

$$\begin{cases}\varepsilon_x=\dfrac{\partial u}{\partial x} & \varepsilon_y=\dfrac{\partial u}{\partial y} & \varepsilon_z=\dfrac{\partial u}{\partial z}\\ \gamma_{xy}=\dfrac{\partial u}{\partial y}+\dfrac{\partial v}{\partial x}=\gamma_{yx} & \gamma_{yz}=\dfrac{\partial v}{\partial z}+\dfrac{\partial w}{\partial y}=\gamma_{zy} & \gamma_{zx}=\dfrac{\partial w}{\partial x}+\dfrac{\partial u}{\partial z}=\gamma_{xz}\end{cases} \tag{6.66}$$

写成矩阵形式为

$$\boldsymbol{\varepsilon}=\boldsymbol{L}\boldsymbol{u}=\boldsymbol{A}^{\mathrm{T}}\boldsymbol{u}(\text{在 } V \text{ 内}) \tag{6.67}$$

③ 物理方程——应力-应变关系

弹性力学中应力-应变之间的转换关系也称弹性关系，考虑各向同性的线弹性材料，应力通过应变的表达式可表示为

$$\boldsymbol{\sigma}=\boldsymbol{D}\boldsymbol{\varepsilon} \tag{6.68}$$

其中

$$\boldsymbol{D}=\frac{E(1-v)}{(1+v)(1-2v)}\begin{bmatrix}1 & \dfrac{v}{1-v} & \dfrac{v}{1-v} & 0 & 0 & 0\\ & 1 & \dfrac{v}{1-v} & 0 & 0 & 0\\ & & 1 & 0 & 0 & 0\\ & \text{对} & & \dfrac{1-2v}{2(1-v)} & 0 & 0\\ & & \text{称} & & \dfrac{1-2v}{2(1-v)} & 0\\ & & & & & \dfrac{1-2v}{2(1-v)}\end{bmatrix} \tag{6.69}$$

称为弹性矩阵，它完全取决于弹性体材料的弹性模量E和泊桑比ν。

④ 力的边界条件

弹性体V的全部边界记为S。一部分边界上已知外力为$\bar{T}_x,\bar{T}_y,\bar{T}_z$，称为力的边界条件，这部分用$S_\sigma$表示，另一部分边界上弹性体的位移$\bar{u}_x,\bar{v}_y,\bar{w}_z$已知，称为几何边界条件或位移边界条件，这部分用$S_u$表示。两部分构成全部边界，即$S=S_\sigma+S_u$。

弹性体在边界上单位面积的内力为 T_x, T_y, T_z，在边界 S_σ 上已知弹性体单位面积上作用的面积力为 $\overline{T}_x, \overline{T}_y, \overline{T}_z$，根据平衡原理，应有

$$T_x = \overline{T}_x, \quad T_y = \overline{T}_y, \quad T_z = \overline{T}_z \tag{6.70}$$

设边界外法线为 N，其方向余弦为 n_x, n_y, n_z，则边界上弹性体的内力可由下式确定

$$\begin{cases} T_x = n_x\sigma_x + n_y\tau_{yx} + n_z\tau_{zx} \\ T_y = n_x\tau_{xy} + n_y\sigma_y + n_z\tau_{zy} \\ T_z = n_x\tau_{xz} + n_y\tau_{yz} + n_z\sigma_z \end{cases} \tag{6.71}$$

将式(6.70)与式(6.71)写成矩阵形式

$$\boldsymbol{T} = \overline{\boldsymbol{T}} \quad (\text{在 } S_\sigma \text{ 上}), \text{其中 } \boldsymbol{T} = \boldsymbol{n}\boldsymbol{\sigma} \tag{6.72}$$

$$\boldsymbol{n} = \begin{bmatrix} n_x & 0 & 0 & n_y & 0 & n_z \\ 0 & n_y & 0 & n_x & n_z & 0 \\ 0 & 0 & n_z & 0 & n_y & n_x \end{bmatrix} \tag{6.73}$$

⑤ 几何边界条件

在 S_u 上弹性体的位移已知为 $\overline{u}, \overline{v}, \overline{w}$，即有

$$u = \overline{u}, \quad v = \overline{v}, \quad w = \overline{w}$$

写成矩阵形式

$$\boldsymbol{u} = \overline{\boldsymbol{u}} \quad (\text{在 } S_u \text{ 上}) \tag{6.74}$$

以上是描述三维弹性力学问题和边界条件的基本方程，下面将其归并到一起，给出其矩阵表达形式如下

$$\left.\begin{array}{ll} \text{平衡方程} & \boldsymbol{A}\boldsymbol{\sigma} + \overline{\boldsymbol{f}} = \boldsymbol{0} \quad (\text{在 } V \text{ 内}) \\ \text{几何方程} & \boldsymbol{\varepsilon} = \boldsymbol{L}\boldsymbol{u} \quad (\text{在 } V \text{ 内}) \\ \text{物理方程} & \boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\varepsilon} \quad (\text{在 } V \text{ 内}) \\ \text{边界条件} & \boldsymbol{n}\boldsymbol{\sigma} = \overline{\boldsymbol{T}} \quad (\text{在 } S_\sigma \text{ 内}) \\ & \boldsymbol{u} = \overline{\boldsymbol{u}} \quad (\text{在 } S_u \text{ 内}) \end{array}\right\} \tag{6.75}$$

⑥ 弹性体的应变能和余能

单位体积的应变能(应变能密度)可以由下式计算得到

$$U(\boldsymbol{\varepsilon}) = \frac{1}{2}\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{\varepsilon} \tag{6.76}$$

显然，应变能是正定函数，只有当弹性体内所有点均无应变时，应变能才为 0。

单位体积的余能(余能密度)可以由下式计算得到

$$V(\boldsymbol{\sigma}) = \frac{1}{2}\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{D}^{-1}\boldsymbol{\sigma} = \frac{1}{2}\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{\sigma} \tag{6.77}$$

余能也是正定函数，在线性弹性力学中，应变能等于余能。

弹性力学基本方程也可以用笛卡儿张量符号表示。在直角坐标系 x_1, x_2, x_3 中，应力张量和应变张量都是对称的二阶张量，分别用 σ_{ij} 和 ε_{ij} 表示，且有 $\sigma_{ij} = \sigma_{ji}, \varepsilon_{ij} = \varepsilon_{ji}$，而位移张量、体积张量、面积张量等均为一阶张量，分别用 $u_i, \overline{f}_i, \overline{T}_i$ 表示，下面分别给出张量形式的弹性力学基本方程及边界条件的表达式。

① 平衡方程

$$\sigma_{ij,j}+\bar{f}_i=0 \quad (\text{在 } V \text{ 内}) \tag{6.78}$$

式中下标“j”表示对独立坐标 $x_j(j=1,2,3)$ 求偏导数，其展开式为

$$\begin{cases}\dfrac{\partial\sigma_{11}}{\partial x_1}+\dfrac{\partial\sigma_{12}}{\partial x_2}+\dfrac{\partial\sigma_{13}}{\partial x_3}+\bar{f}_1=0\\ \dfrac{\partial\sigma_{21}}{\partial x_1}+\dfrac{\partial\sigma_{22}}{\partial x_2}+\dfrac{\partial\sigma_{23}}{\partial x_3}+\bar{f}_2=0\\ \dfrac{\partial\sigma_{31}}{\partial x_1}+\dfrac{\partial\sigma_{32}}{\partial x_2}+\dfrac{\partial\sigma_{33}}{\partial x_3}+\bar{f}_3=0\end{cases} \tag{6.79}$$

坐标及应力张量见图 6.12。

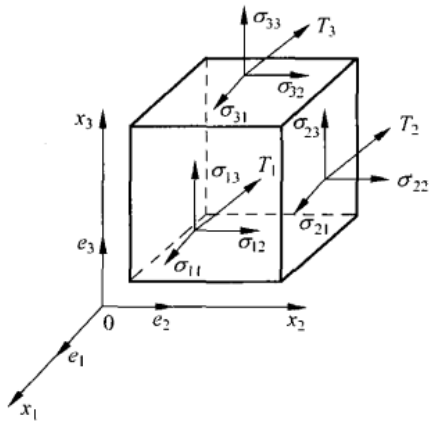


图 6.12 应力张量及其分量

② 几何方程

$$\varepsilon_{ij}=\frac{1}{2}(u_{i,j}+u_{j,i}) \quad (\text{在 } V \text{ 内}) \tag{6.80}$$

其展开式是

$$\begin{cases}\varepsilon_{11}=\dfrac{\partial u_1}{\partial x_1}\\ \varepsilon_{22}=\dfrac{\partial u_2}{\partial x_2}\\ \varepsilon_{33}=\dfrac{\partial u_3}{\partial x_3}\\ \varepsilon_{12}=\dfrac{1}{2}\left(\dfrac{\partial u_1}{\partial x_2}+\dfrac{\partial u_2}{\partial x_1}\right)=\varepsilon_{21}\\ \varepsilon_{23}=\dfrac{1}{2}\left(\dfrac{\partial u_2}{\partial x_3}+\dfrac{\partial u_3}{\partial x_2}\right)=\varepsilon_{32}\\ \varepsilon_{31}=\dfrac{1}{2}\left(\dfrac{\partial u_3}{\partial x_1}+\dfrac{\partial u_1}{\partial x_3}\right)=\varepsilon_{13}\end{cases} \tag{6.81}$$

与式(6.66)相比较，当 x_1,x_2,x_3 就是笛卡儿坐标时，

$$\varepsilon_{11}=\varepsilon_x,\quad \varepsilon_{22}=\varepsilon_y,\quad \varepsilon_{33}=\varepsilon_z,\quad \varepsilon_{12}=\frac{1}{2}\gamma_{xy},\quad \varepsilon_{23}=\frac{1}{2}\gamma_{yz},\quad \varepsilon_{31}=\frac{1}{2}\gamma_{zx}$$

③ 物理方程

广义虎克定律假定每个应力分量与各应变分量成正比，可以用张量形式表示为

$$\boldsymbol{\sigma}_{ij}=D_{ijkl}\boldsymbol{\varepsilon}_{kl} \quad (\text{在 } V \text{ 内}) \tag{6.82}$$

比例常数 D_{ijkl} 称为弹性常数，共 81 个。应力张量与应变张量均为对称张量，因此 D_{ijkl} 的两个前指标具有对称性，两个后指标也具有对称性，即 $D_{ijkl}=D_{jikl}$，$D_{ijkl}=D_{ijlk}$。特别是，当变形过程是绝热或等温过程时，还有 $D_{ijkl}=D_{klij}$。对上述情况，一般线弹性材料，即在不同方向具有不同弹性性质的材料，81 个弹性常数有 21 个是独立的。对于各向同性的线弹性材料，独立的弹性常数只有两个，即拉梅常数 G 和 λ 或弹性模量 E 和泊桑比 ν，这些弹性常数的关系为

$$G=\frac{E}{2(1+\nu)},\quad \lambda=\frac{E\nu}{(1+\nu)(1-2\nu)} \tag{6.83}$$

此时弹性张量可以简化为

$$D_{ijkl} = 2G\delta_{ik}\delta_{jl} + \lambda\delta_{ij}\delta_{kl} \tag{6.84}$$

广义虎克定律可以表示为

$$\sigma_{ij} = 2G\varepsilon_{ij} + \lambda\delta_{ij}\varepsilon_{kk} \tag{6.85}$$

其中

$$\delta_{ij} = \begin{cases} 1, & \text{当 } i = j \\ 0, & \text{当 } i \neq j \end{cases}$$

其展开式为

$$\begin{cases} \sigma_{11} = 2G\varepsilon_{11} + \lambda(\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}) \\ \sigma_{22} = 2G\varepsilon_{22} + \lambda(\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}) \\ \sigma_{33} = 2G\varepsilon_{33} + \lambda(\varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}) \\ \sigma_{12} = 2G\varepsilon_{12}, \sigma_{23} = 2G\varepsilon_{23}, \sigma_{31} = 2G\varepsilon_{31} \end{cases} \tag{6.86}$$

④ 力的边界条件

由 $T_i = \overline{T}_i$(在 S_σ 上),其中 $T_i = \sigma_{ij}n_j$,n_j 是边界外法线 N 的三个方向余弦,将其代入,其展开式为

$$\begin{cases} \sigma_{11}n_1 + \sigma_{12}n_2 + \sigma_{13}n_3 = \overline{T}_1 \\ \sigma_{21}n_1 + \sigma_{22}n_2 + \sigma_{23}n_3 = \overline{T}_2 \quad (\text{在 } S_\sigma \text{ 上}) \\ \sigma_{31}n_1 + \sigma_{32}n_2 + \sigma_{33}n_3 = \overline{T}_3 \end{cases} \tag{6.87}$$

⑤ 位移边界条件

$$u_i = \overline{u}_i \quad (\text{在 } S_u \text{ 上}) \tag{6.88}$$

以上是弹性力学基本方程用笛卡儿张量符号表示的形式。

⑥ 应变能和余能

单位体积应变能

$$U(\varepsilon_{mn}) = \frac{1}{2}D_{ijkl}\varepsilon_{ij}\varepsilon_{kl} \tag{6.89}$$

单位体积余能

$$V(\sigma_{mn}) = \frac{1}{2}C_{ijkl}\sigma_{ij}\sigma_{kl} \tag{6.90}$$

6.6 有限元方法原理

6.6.1 三角形单元的有限元格式

本节以平面为例介绍有限元方法的原理。由于三角形单元对复杂边界有较强的适应力,因此,讨论三角形单元的有限元格式。

用三角形对二维域进行剖分的原理可参见图 6.13。在边界上以若干线段近似原来的曲线。随着单元增多,这种拟合就越精确。需要对三角形的顶点进行编号,典型的三节点三角形单元其节点编号的规则是:以 i,j,m 为序,以顺时针方向编码为正向。每个节点有 2

个位移分量(参见图 6.14)。

$$\boldsymbol{a}_i = \begin{Bmatrix} u_i \\ v_i \end{Bmatrix}, \quad \boldsymbol{a}_j = \begin{Bmatrix} u_j \\ v_j \end{Bmatrix}, \quad \boldsymbol{a}_m = \begin{Bmatrix} u_m \\ v_m \end{Bmatrix}$$

每个单元有 6 个节点位移即 6 个节点自由度,表示为

$$\boldsymbol{a}^e = \begin{Bmatrix} a_i \\ a_j \\ a_m \end{Bmatrix} = [u_i \quad v_i \quad u_j \quad v_j \quad u_m \quad v_m]^T$$

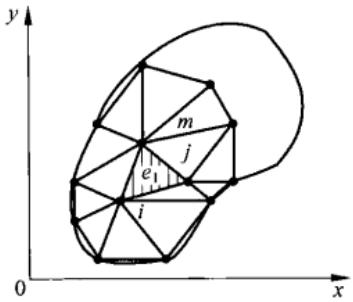


图 6.13　二维域三角形离散

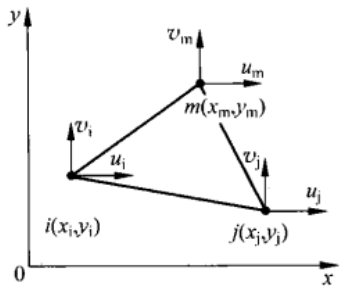


图 6.14　三节点三角形单元

1. 单元的位移模式及插值函数

在有限元方法中,单元的位移模式也称位移函数,一般采用多项式作为近似函数,因为其运算简便,而且随着项数增加,可以逼近任何光滑的函数曲线段。一般可以从低到高逐步增加。三节点三角形单元位移函数一般为一次多项式,其形式如下

$$\begin{aligned} u &= \beta_1 + \beta_2 x + \beta_3 y \\ v &= \beta_4 + \beta_5 x + \beta_6 y \end{aligned} \tag{6.91}$$

单元内的位移是坐标(x,y)的线性函数,$\beta_1 \sim \beta_6$ 是待定系数,称为广义坐标。6 个广义坐标可由单元的 6 个节点位移来表示。将 i,j,m 分别作为下标代入式(6.91),就可得到

x 方向的位移

$$\begin{cases} u_i = \beta_1 + \beta_2 x_i + \beta_3 y_i \\ u_j = \beta_1 + \beta_2 x_j + \beta_3 y_j \\ u_m = \beta_1 + \beta_2 x_m + \beta_3 y_m \end{cases} \tag{6.92}$$

y 方向的位移

$$\begin{cases} v_i = \beta_4 + \beta_5 x_i + \beta_6 y_i \\ v_j = \beta_4 + \beta_5 x_j + \beta_6 y_j \\ v_m = \beta_4 + \beta_5 x_m + \beta_6 y_m \end{cases} \tag{6.93}$$

不难从式(6.77)、式(6.78)求得待定系数 $\beta_1 \sim \beta_6$。因为其系数行列式为

$$D = \begin{vmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_m & y_m \end{vmatrix} = 2A \quad (A \text{ 是三角形面积})$$

$$\beta_1=\frac{1}{D}\begin{vmatrix} u_i & x_i & y_i \\ u_j & x_j & y_j \\ u_m & x_m & y_m \end{vmatrix}=\frac{1}{2A}(a_iu_i+a_ju_j+a_mu_m)$$

$$\beta_2=\frac{1}{D}\begin{vmatrix} 1 & u_i & y_i \\ 1 & u_j & y_j \\ 1 & u_m & y_m \end{vmatrix}=\frac{1}{2A}(b_iu_i+b_ju_j+b_mu_m)$$

$$\beta_3=\frac{1}{D}\begin{vmatrix} 1 & x_i & u_i \\ 1 & x_j & u_j \\ 1 & x_m & u_m \end{vmatrix}=\frac{1}{2A}(c_iu_i+c_ju_j+c_mu_m)$$

$$\beta_4=\frac{1}{D}\begin{vmatrix} v_i & x_i & y_i \\ v_j & x_j & y_j \\ v_m & x_m & y_m \end{vmatrix}=\frac{1}{2A}(a_iv_i+a_jv_j+a_mv_m)$$

$$\beta_5=\frac{1}{D}\begin{vmatrix} 1 & v_i & y_i \\ 1 & v_j & y_j \\ 1 & v_m & y_m \end{vmatrix}=\frac{1}{2A}(b_iv_i+b_jv_j+b_mv_m)$$

$$\beta_6=\frac{1}{D}\begin{vmatrix} 1 & x_i & v_i \\ 1 & x_j & v_j \\ 1 & x_m & v_m \end{vmatrix}=\frac{1}{2A}(c_iv_i+c_jv_j+c_mv_m)$$

其中

$$\begin{cases} a_i=\begin{vmatrix} x_j & y_j \\ x_m & y_m \end{vmatrix}=x_jy_m-x_my_j \\ b_i=-\begin{vmatrix} 1 & y_j \\ 1 & y_m \end{vmatrix}=y_j-y_m \quad (i,j,m) \\ c_i=\begin{vmatrix} 1 & x_j \\ 1 & x_m \end{vmatrix}=-x_j-x_m \end{cases} \tag{6.94}$$

式(6.94)中(i,j,m)表示下标轮换，如$i\to j,j\to m,m\to i$。这样，求得的广义坐标代入式(6.91)，可得到

$$\begin{cases} u=N_iu_i+N_ju_j+N_mu_m \\ v=N_iv_i+N_jv_j+N_mv_m \end{cases} \tag{6.95}$$

其中

$$N_i=\frac{1}{2A}(a_i+b_ix+c_iy) \quad (i,j,m) \tag{6.96}$$

N_i,N_j,N_m称为单元的插值函数或形函数，它是坐标的一次函数。a_i,b_i,c_i；a_j,b_j,c_j；a_m,b_m,c_m均是常数，其值取决于单元的三个节点的坐标。将式(6.95)用矩阵表示，可得到

$$\boldsymbol{u}=\begin{Bmatrix} u \\ v \end{Bmatrix}=\begin{bmatrix} N_i & 0 & N_j & 0 & N_m & 0 \\ 0 & N_i & 0 & N_j & 0 & N_m \end{bmatrix}[u_i \quad v_i \quad u_j \quad v_j \quad u_m \quad v_m]^{\mathrm{T}}$$

$$=[\boldsymbol{I}N_i \quad \boldsymbol{I}N_j \quad \boldsymbol{I}N_m]\begin{Bmatrix} \boldsymbol{a}_i \\ \boldsymbol{a}_j \\ \boldsymbol{a}_m \end{Bmatrix}=[\boldsymbol{N}_i \quad \boldsymbol{N}_j \quad \boldsymbol{N}_m]\boldsymbol{a}^e=\boldsymbol{N}\boldsymbol{a}^e \tag{6.97}$$

$\boldsymbol{N}$ 称为插值函数矩阵或形函数矩阵。它具有以下性质：

(1) $N_i(x_i,y_i)=1, N_j(x_j,y_j)=N_m(x_m,y_m)=0$，即在节点 i，应有 $u=u_i$，用数学表达式表示，则为

$$N_i(x_j,y_j)=\delta_{ij}=\begin{cases}1 & \text{当 } j=i\\ 0 & \text{当 } j\neq i\end{cases}\quad (i,j,m) \tag{6.98}$$

(2) 在单元中任一点各插值函数之和等于 1，即 $N_i+N_j+N_m=1$。其物理意义是：若单元发生刚体位移 u_0，则单元内所有节点均有位移 u_0，因为

$$u=N_iu_i+N_ju_j+N_mu_m=(N_i+N_j+N_m)u_0=u_0$$

所以必须保证 $N_i+N_j+N_m=1$。

(3) 插值函数是线性的，在单元内部及边界上的位移也是线性的，节点上的位移值唯一地确定了相邻单元公共节点的位移，这样相邻单元在公共边界上位移是连续的。

2. 应变矩阵和应力矩阵

确定了单元位移后，利用几何方程和物理方程易于求得单元的应变和应力。

$$\begin{aligned}\boldsymbol{\varepsilon}&=\begin{Bmatrix}\varepsilon_x\\ \varepsilon_y\\ \gamma_{xy}\end{Bmatrix}=\boldsymbol{Lu}=\boldsymbol{LNa}^e=\boldsymbol{L}[\boldsymbol{N}_i\quad \boldsymbol{N}_j\quad \boldsymbol{N}_m]\boldsymbol{a}^e\\ &=[\boldsymbol{B}_i\quad \boldsymbol{B}_j\quad \boldsymbol{B}_m]\boldsymbol{a}^e=\boldsymbol{Ba}^e\end{aligned} \tag{6.99}$$

其中 $\boldsymbol{B}$ 称为应变矩阵，$\boldsymbol{L}$ 是平面问题的微分算子。三节点单元的应变矩阵是

$$\boldsymbol{B}=[\boldsymbol{B}_i\quad \boldsymbol{B}_j\quad \boldsymbol{B}_m]=\frac{1}{2A}\begin{bmatrix}b_i & 0 & b_j & 0 & b_m & 0\\ 0 & c_i & 0 & c_j & 0 & c_m\\ c_i & b_i & c_j & b_j & c_m & b_m\end{bmatrix} \tag{6.100}$$

单元应力可以根据物理方程求得，即

$$\boldsymbol{\sigma}=\begin{Bmatrix}\sigma_x\\ \sigma_y\\ \tau_{xy}\end{Bmatrix}=\boldsymbol{D\varepsilon}=\boldsymbol{DBa}^e=\boldsymbol{Sa}^e \tag{6.101}$$

其中 $\boldsymbol{S}$ 称为应力矩阵

$$\boldsymbol{S}=\boldsymbol{DB}=\boldsymbol{D}[\boldsymbol{B}_i\quad \boldsymbol{B}_j\quad \boldsymbol{B}_m]=[\boldsymbol{S}_i\quad \boldsymbol{S}_j\quad \boldsymbol{S}_m] \tag{6.102}$$

$\boldsymbol{S}$ 的分块矩阵为

$$\boldsymbol{S}_i=\boldsymbol{DB}_i=\frac{E_0}{2(1-\nu_0^2)A}\begin{bmatrix}b_i & \nu_0c_i\\ \nu_0b_i & c_i\\ \dfrac{1-\nu_0}{2}c_i & \dfrac{1-\nu_0}{2}b_i\end{bmatrix}\quad (i,j,m) \tag{6.103}$$

其中 E_0,ν_0 为材料常数，对于平面应力问题，$E_0=E,\nu_0=\nu$，对于平面应变问题

$$E_0=E/(1-\nu^2),\nu_0=\nu/(1-\nu)$$

6.6.2 基于最小位能原理的有限元方程

最小位能原理基于虚位移原理。首先考虑平衡方程

$$\sigma_{ij,j} + \bar{f}_i = 0 \quad (\text{在 } V \text{ 内}) \tag{6.104}$$

以及力的边界条件

$$\sigma_{ij} n_j - \bar{T}_i = 0 \quad (\text{在 } S_\sigma \text{ 上}) \tag{6.105}$$

我们可以利用式(6.54)建立与它们等效的积分形式。现在的平衡条件相当于$A(\boldsymbol{u})=0$，力的边界条件相当于 $B(\boldsymbol{u})=0$。权函数可分别取真实位移的变分 δu_i 及其边界值(取负值)，这不失一般性。这样就可得到上述平衡方程的等效积分

$$\int_V \delta u_i (\sigma_{ij,j} + \bar{f}_i) \mathrm{d}V - \int_{S_\sigma} (\sigma_{ij} n_j - \bar{T}_i) \mathrm{d}S = 0 \tag{6.106}$$

注意到应力张量 σ_{ij} 是对称的，且考虑到真实位移的变分所导致的在给定位移边界条件的 S_u 上 $\delta u_i=0$，以及在体积内部满足几何协调条件 $\frac{1}{2}(\delta u_{i,j}+\delta_{j,i})=\delta\varepsilon_{ij}$，则上式中

$$\begin{aligned}\int_V \delta u_i \sigma_{ij,j} \mathrm{d}V &= -\int_V \frac{1}{2}(\delta u_{i,j} + \delta u_{j,i})\sigma_{ij,j} \mathrm{d}V + \int_{S_\sigma} \delta u_i \sigma_{ij} n_j \mathrm{d}S \\ &= -\int_V \delta\varepsilon_{ij}\sigma_{ij} \mathrm{d}V + \int_{S_\sigma} \delta u_i \sigma_{ij} n_j \mathrm{d}S\end{aligned}$$

代入式(6.106)可得到

$$\int_V (\delta\varepsilon_{ij}\sigma_{ij} - \delta u_i \bar{f}_i) \mathrm{d}V - \int_{S_\sigma} \delta u_i \bar{T}_i \mathrm{d}S = 0 \tag{6.107}$$

注意，上式体积积分的第一项称为变形体内的应力在虚应变上所做的功，即内力的虚功；第二项及面积积分分别称为体积力和面积力在虚位移上所做的功，即外力的虚功。内力的虚功与外力的虚功两者之和为 0，称为虚功原理。虚功是外力和内力分别在虚位移及与之相协调的虚应变上所做的功，从而得到虚位移原理，其力学意义是：如果力系(包括内力 σ 和外力 $\bar{f}$ 及 $\bar{T}$)是平衡的(满足方程式(6.104)及式(6.105))，则它们在虚位移和虚应变上所做功的总和为 0。

若利用弹性力学物理方程式(6.82)，将其代入式(6.107)第一项，并考虑 D_{ijkl} 是对称张量，则有

$$(\delta\varepsilon_{ij}) D_{ijkl}\varepsilon_{kl} = \delta\left(\frac{1}{2} D_{ijkl}\varepsilon_{ij}\varepsilon_{kl}\right) = \delta U(\varepsilon_{mn})$$

可见式(6.107)积分第一项就是单位体积应变能的变分。

在线性弹性力学中，假定体积力 $\bar{f}_i$ 和边界上面力 $\bar{T}_i$ 的大小和方向都是不变的，即可从位势函数 $\phi(u_i)$ 和 $\psi(u_i)$ 导出，则有

$$-\delta\phi(u_i) = \bar{f}_i \delta u_i \qquad -\delta\psi(u_i) = \bar{T}_i \delta u_i$$

将其代入式(6.107)，可得

$$\delta \boldsymbol{\Pi}_p = 0 \tag{6.108}$$

其中

$$\begin{aligned}\boldsymbol{\Pi}_p = \boldsymbol{\Pi}_p(\varepsilon_{ij}, u_i) &= \int_V [U(\varepsilon_{ij}) + \phi(u_i)] \mathrm{d}V + \int_{S_\sigma} \psi(u_i) \mathrm{d}S \\ &= \int_V \left(\frac{1}{2} D_{ijkl}\varepsilon_{ij}\varepsilon_{kl} - \bar{f}_i u_i\right) \mathrm{d}V - \int_{S_\sigma} \bar{T}_i u_i \mathrm{d}S\end{aligned} \tag{6.109}$$

它是系统的总位能，也是弹性体变形位能和外力位能之和。式(6.108)表明，在所有区域内，

满足几何关系式(6.81)和边界上满足定位条件式(6.88)的可能位移中，真实位移使系统的总位能取驻值。

将上述最小位能原理用于平面弹性体，并采用矩阵表示，可得

$$\boldsymbol{\Pi}_{\mathrm{p}}=\int_{a}\frac{1}{2}\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{\varepsilon}t\mathrm{d}x\mathrm{d}y-\int_{a}\boldsymbol{u}^{\mathrm{T}}\bar{\boldsymbol{f}}t\mathrm{d}x\mathrm{d}y-\int_{S_{\sigma}}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{T}t\mathrm{d}S \tag{6.110}$$

其中

t 是二维体内厚度；

$\boldsymbol{f}$ 是作用在二维体上的面积力；

$\boldsymbol{T}$ 是作用在二维体边界上的面积力。

对于离散模型，系统位能是各单元位能的和，将式(6.97)、式(6.99)代入式(6.110)，即得到离散模型的总位能

$$\boldsymbol{\Pi}_{\mathrm{p}}=\sum_{e}\left(\boldsymbol{a}^{e\mathrm{T}}\int_{\Omega^{e}}\frac{1}{2}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}t\mathrm{d}x\mathrm{d}y\boldsymbol{a}^{e}\right)-\sum_{e}\left(\boldsymbol{a}^{e\mathrm{T}}\int_{\Omega^{e}}\boldsymbol{N}^{\mathrm{T}}\bar{\boldsymbol{f}}t\mathrm{d}x\mathrm{d}y\right)-\sum_{e}\left(\boldsymbol{a}^{e\mathrm{T}}\int_{S_{\sigma}^{e}}\boldsymbol{N}^{\mathrm{T}}\boldsymbol{T}t\mathrm{d}S\right) \tag{6.111}$$

将单元节点位移 $\boldsymbol{a}^{e}$ 用结构节点位移 $\boldsymbol{a}$ 表示，即 $\boldsymbol{a}^{e}=\boldsymbol{G}\boldsymbol{a}$，其中

$$\boldsymbol{a}=[u_1\quad v_1\quad u_2\quad v_2\quad\cdots\quad u_i\quad v_i\quad\cdots\quad u_n\quad v_n]^{\mathrm{T}}$$

$$\boldsymbol{G}_{6\times 2n}=\begin{array}{c}\begin{matrix}1&2&\cdots&2i-1&2i&\cdots&2m-1&2m&\cdots&2j-1&2j&\cdots&2n\end{matrix}\\\begin{bmatrix}0&0&\cdots&1&0&\cdots&0&0&\cdots&0&0&\cdots&0\\0&0&\cdots&0&1&\cdots&0&0&\cdots&0&0&\cdots&0\\0&0&\cdots&0&0&\cdots&0&0&\cdots&1&0&\cdots&0\\0&0&\cdots&0&0&\cdots&0&0&\cdots&0&1&\cdots&0\\0&0&\cdots&0&0&\cdots&1&0&\cdots&0&0&\cdots&0\\0&0&\cdots&0&0&\cdots&0&1&\cdots&0&0&\cdots&0\end{bmatrix}\end{array}$$

式中的 n 为结构节点数。令 $\int_{\Omega^{e}}\frac{1}{2}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}t\mathrm{d}x\mathrm{d}y=\boldsymbol{K}^{e}$(称为单元刚度降)，并令 $\int_{\Omega^{e}}\boldsymbol{N}^{\mathrm{T}}\bar{\boldsymbol{f}}t\mathrm{d}x\mathrm{d}y=\boldsymbol{P}_{\mathrm{b}}^{e}$，$\int_{S_{\sigma}^{e}}\boldsymbol{N}^{\mathrm{T}}\boldsymbol{T}t\mathrm{d}S=\boldsymbol{P}_{\mathrm{S}}^{e}$，且 $\boldsymbol{P}^{e}=\boldsymbol{P}_{\mathrm{b}}^{e}+\boldsymbol{P}_{\mathrm{S}}^{e}$ 称为单元等效节点载荷列阵，将它们代入式(6.111)，离散形式的总位能可表示为

$$\boldsymbol{\Pi}_{\mathrm{p}}=\boldsymbol{a}^{\mathrm{T}}\frac{1}{2}\sum_{e}(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{K}^{e}\boldsymbol{G})\boldsymbol{a}-\boldsymbol{a}^{\mathrm{T}}\sum_{e}(\boldsymbol{G}^{\mathrm{T}}\boldsymbol{P}^{e}) \tag{6.112}$$

若令 $\boldsymbol{K}=\sum_{e}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{K}^{e}\boldsymbol{G}$，称为结构体的整体刚度矩阵，$\boldsymbol{P}=\sum_{e}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{P}^{e}$ 称为结构节点载荷列阵，式(6.112)可写为

$$\boldsymbol{\Pi}_{\mathrm{p}}=\frac{1}{2}\boldsymbol{a}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{a}+\boldsymbol{a}^{\mathrm{T}}\boldsymbol{P} \tag{6.113}$$

离散形式的总位能 $\boldsymbol{\Pi}_{\mathrm{p}}$ 的未知变量是结构的节点位移 $\boldsymbol{a}$，根据变分原理，泛函 $\boldsymbol{\Pi}_{\mathrm{p}}$ 取驻值的条件是其一阶变分为0，即 $\delta\boldsymbol{\Pi}_{\mathrm{p}}=\frac{\partial\boldsymbol{\Pi}_{\mathrm{p}}}{\partial\boldsymbol{a}}=0$，这样就可得到有限元求解方程

$$\boldsymbol{K}\boldsymbol{a}=\boldsymbol{P} \tag{6.114}$$

从结构体整体刚度矩阵和结构节点载荷矩阵的定义可知，$\boldsymbol{K}$ 和 $\boldsymbol{P}$ 都是由单元刚度矩阵和单元等效节点载荷列阵集合而成，而 $\boldsymbol{G}$ 为转换矩阵。$\boldsymbol{P}^{e}$ 的特性依赖于载荷特性，为此，我

们需要进一步分析 $\boldsymbol{K}^e$。

单元刚度矩阵的物理意义可以由最小位能原理加以说明。考虑一个单元的平衡，可以写成 $\boldsymbol{K}^e\boldsymbol{a}^e=\boldsymbol{P}^e$，若记

$$\boldsymbol{a}^e=[u_i \quad v_i \quad u_j \quad v_j \quad u_m \quad v_m]^T=[a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6]^T$$

$$\boldsymbol{P}^e=[P_{ix} \quad P_{iy} \quad P_{jx} \quad P_{jy} \quad P_{mx} \quad P_{my}]^T=[P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5 \quad P_6]^T$$

则式(6.113)的展开式是

$$\begin{bmatrix} K_{11} & K_{12} & \cdots & K_{16} \\ K_{21} & K_{22} & \cdots & K_{26} \\ \vdots & \vdots & & \vdots \\ K_{61} & K_{62} & \cdots & K_{66} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{bmatrix} \tag{6.115}$$

这是单元节点平衡方程，方程左端表示单元节点内力，右边是外载荷。若令 $a_1=1(u_i=1)$，而 $a_2,a_3,\cdots$均为 0，为保证该节点平衡需要在相应位移方向施加载荷的大小，其表达式为

$$\begin{bmatrix} K_{11} \\ K_{21} \\ \vdots \\ K_{61} \end{bmatrix}_{a_1=1} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_6 \end{bmatrix}$$

可以证明，单元刚度矩阵具有如下特性：

① 对称性：单元刚度矩阵是对称矩阵；

② 奇异性：单元刚度矩阵是奇异的，它不存在逆矩阵，它的秩是 3；

③ 主元为正：其物理意义是，要使节点位移 $a_i=1$，施加在 a_i 方向的节点力必须与位移 a_i 的方向相同。

单元刚度矩阵的奇异特性说明，由它们集成的整体结构刚度矩阵 $\boldsymbol{K}$ 是奇异的，为消除奇异，必须附加限制刚体位移的约束条件，才能求出在某种载荷下的节点位移。一般采用几何边界约束条件的办法来实现，这里就不再进一步讨论。详细内容请参考相关的著作或文献。

6.6.3 广义坐标有限元法的一般格式

二维或三维连续体离散为有限元的集合体时，要求单元具有简单而规则的几何形状，以便于仿真计算。常用的二维单元为三角形或矩形，三维单元有四面体(三角锥)、五面体或平行六面体。同样形状的单元可以有不同的单元节点数，如二维三角形单元有 3 节点、6 节点、10 节点等。如何选择合适的单元进行计算，涉及求解问题的类型、计算精度的要求，以及计算速度等。

前面几节讨论了有限元法的基本原理，本节介绍广义坐标有限元法的一般格式。

(1) 单元位移函数的选择

单元位移函数的选择是十分重要的，它对计算精度、速度有重要的影响。单元中的位移模式一般采用以广义坐标 $\boldsymbol{\beta}$ 为待定参数的有限项多项式作为近似函数，其选择应遵循以下基本原则：

① 广义坐标是由节点场变量确定的，因此它的个数应与节点自由度数相等。

例如，3节点三角形单元的节点位移自由度为6，广义坐标数也应是6，从而两个方向的位移 u 和 v 要各取三项多项式。类似地，4节点的矩形单元位移自由度为8，广义坐标数也应是8，位移 u 和 v 要各取四项多项式作为近似函数。

② 多项式的常数项和坐标的一次项必须完备。位移模式中的常数项和一次项反映了单元刚体位移和常应变特性，当单元数趋于无穷时，单元缩小趋于一点，此时单元应变应趋于常应变，完备的常数项和坐标的一次项是保证这一特性的条件。

③ 多项式的选取一般由低到高，且尽量选取完全多项式以提高单元精度。

例如，每边为2个端节点的应保证一次完全多项式，每边有3个节点时应取二次完全多项式。若由于项数限制不能选取完全多项式时，多项式的坐标应满足对称性，并且一个坐标方向的次数不应超过完全多项式的次数。

(2) 广义坐标有限元的一般格式

本节以3节点三角形单元为例给出广义坐标有限元的一般格式。

① 以广义坐标 $\boldsymbol{\beta}$ 为待定参数，单元内位移 $\boldsymbol{u}$ 表达式为

$$\boldsymbol{u}=\boldsymbol{\varphi}\boldsymbol{\beta}$$

对于二维问题

$$\boldsymbol{u}=\begin{Bmatrix}u\\v\end{Bmatrix},\quad \boldsymbol{\beta}=[\beta_1\quad \beta_2\quad \cdots]^{\mathrm{T}},\quad \boldsymbol{\varphi}=\begin{bmatrix}\varphi & 0\\0 & \varphi\end{bmatrix}$$

三角形常应变单元(局部节点编码 i,j,m 用1,2,3代替)

$$u=\beta_1+\beta_2 x+\beta_3 y,\quad \boldsymbol{\beta}=[\beta_1\quad \beta_2\quad \cdots\quad \beta_6]^{\mathrm{T}}$$
$$v=\beta_4+\beta_5 x+\beta_6 y,\quad \boldsymbol{\varphi}=[1\quad x\quad y]$$

② 用单元节点位移 $\tilde{\boldsymbol{a}}^e$ 表示广义坐标 $\boldsymbol{\beta}$

$$\boldsymbol{a}^e=[u_1\quad v_1\quad u_2\quad v_2\quad \cdots]^{\mathrm{T}}$$

为便于计算 $\boldsymbol{\beta}$，令

$$\tilde{\boldsymbol{a}}^e=[u_1\quad u_2\quad \cdots,\quad v_1\quad v_2\quad \cdots]^{\mathrm{T}}$$

将 $\boldsymbol{u}=\boldsymbol{\varphi}\boldsymbol{\beta}$ 代入单元节点坐标可得

$$\tilde{\boldsymbol{a}}^e=\boldsymbol{A}\boldsymbol{\beta}$$

对于二维问题

$$\boldsymbol{A}=\begin{bmatrix}\widetilde{A} & 0\\0 & \widetilde{A}\end{bmatrix},\quad \boldsymbol{A}^{-1}=\begin{bmatrix}\widetilde{A}^{-1} & 0\\0 & \widetilde{A}^{-1}\end{bmatrix},\quad \boldsymbol{\beta}=\boldsymbol{A}^{-1}\tilde{\boldsymbol{a}}^e,\tilde{\boldsymbol{a}}^e=[u_1\quad u_2\quad u_3\quad v_1\quad v_2\quad v_3]^{\mathrm{T}}$$

$$\widetilde{\boldsymbol{A}}=\begin{bmatrix}1 & x_1 & y_1\\1 & x_2 & y_2\\1 & x_3 & y_3\end{bmatrix},\quad \widetilde{\boldsymbol{A}}^{-1}=\frac{1}{\det\widetilde{\boldsymbol{A}}}\begin{bmatrix}a_1 & a_2 & a_3\\b_1 & b_2 & b_3\\c_1 & c_2 & c_3\end{bmatrix}$$

其中

$$\det\widetilde{\boldsymbol{A}}=|\widetilde{\boldsymbol{A}}|=2A=a_1+a_2+a_3=b_2c_3-b_3c_2$$

③ 以节点单元位移 $\boldsymbol{a}^e$ 表示单元位移函数 $\boldsymbol{u}$，得到单元插值函数 $\boldsymbol{N}$，将 $\boldsymbol{\beta}=\boldsymbol{A}^{-1}\tilde{\boldsymbol{a}}^e$ 代入

$$\boldsymbol{u}=\boldsymbol{\varphi}\boldsymbol{\beta}$$

可得

$$\boldsymbol{u}=\boldsymbol{\varphi}\boldsymbol{A}^{-1}\tilde{\boldsymbol{a}}^{e}=\widetilde{\boldsymbol{N}}\tilde{\boldsymbol{a}}^{e}$$

对于二维问题

$$\widetilde{\boldsymbol{N}}=\begin{bmatrix}\boldsymbol{N}^{*} & 0\\ 0 & \boldsymbol{N}^{*}\end{bmatrix},\quad \boldsymbol{N}^{*}=\boldsymbol{\varphi}\widetilde{\boldsymbol{A}}^{-1}=[N_1\quad N_2\quad \cdots],\quad \boldsymbol{N}^{*}=\boldsymbol{\varphi}_{1\times 3}\widetilde{\boldsymbol{A}}_{3\times 3}^{-1}=[N_1\quad N_2\quad N_3]$$

$$N_i=\frac{1}{2A}(a_i+b_ix+c_iy)\quad (i=1,2,3),\quad \widetilde{\boldsymbol{N}}=\begin{bmatrix}N_1 & N_2 & N_3 & 0 & 0 & 0\\ 0 & 0 & 0 & N_1 & N_2 & N_3\end{bmatrix}$$

将节点位移$\tilde{\boldsymbol{a}}^{e}$改为一般排列顺序$\boldsymbol{a}^{e}$,则有

$$\boldsymbol{u}=\boldsymbol{N}\boldsymbol{a}^{e},\boldsymbol{N}=[\boldsymbol{N}_1\quad \boldsymbol{N}_2\quad \boldsymbol{N}_3\quad \cdots]$$

对二维

$$\boldsymbol{N}=[\boldsymbol{N}_1\quad \boldsymbol{N}_2\quad \boldsymbol{N}_3]$$

$$\boldsymbol{N}_i=\boldsymbol{I}N_i\quad (i=1,2,3)$$

④ 以单元节点位移$\boldsymbol{a}^{e}$表示单元应变和应力

应变

$$\boldsymbol{\varepsilon}=\boldsymbol{L}\boldsymbol{u}=\boldsymbol{B}(x,y)\boldsymbol{a}^{e},\boldsymbol{B}=\boldsymbol{L}\boldsymbol{N},\boldsymbol{\varepsilon}=[\varepsilon_x\quad \varepsilon_y\quad \gamma_{xy}]^{\mathrm{T}}$$

$$\boldsymbol{L}=\begin{bmatrix}\dfrac{\partial}{\partial \boldsymbol{x}} & 0\\ 0 & \dfrac{\partial}{\partial \boldsymbol{y}}\\ \dfrac{\partial}{\partial \boldsymbol{y}} & \dfrac{\partial}{\partial \boldsymbol{x}}\end{bmatrix}$$

应力　由弹性变形产生的应力

$$\boldsymbol{\sigma}=\boldsymbol{D}\boldsymbol{\varepsilon}=\boldsymbol{D}\boldsymbol{B}\boldsymbol{a}^{e}$$

当有初应力$\boldsymbol{\sigma}_0$和初应变$\boldsymbol{\varepsilon}_0$时,应力的一般形式是

$$\boldsymbol{\sigma}=\boldsymbol{D}(\boldsymbol{\varepsilon}-\boldsymbol{\varepsilon}_0)+\boldsymbol{\sigma}_0=\boldsymbol{D}\boldsymbol{B}\boldsymbol{a}^{e}-\boldsymbol{D}\boldsymbol{\varepsilon}_0+\boldsymbol{\sigma}_0$$

⑤ 用最小位能原理建立离散体系的节点平衡方程

系统总位能的离散形式

$$\boldsymbol{\Pi}_{\mathrm{p}}=\sum_{e}\int_{V_e}[U+\phi]\mathrm{d}V+\int_{S_\sigma^e}\psi\mathrm{d}S$$

将单元节点的应变、应力表达式代入,并将单元节点位移$\boldsymbol{a}^{e}$用结构节点位移$\boldsymbol{a}$表示,即$\boldsymbol{a}^{e}=\boldsymbol{G}\boldsymbol{a}$,则上式可写成

$$\boldsymbol{\Pi}_{\mathrm{p}}=\boldsymbol{a}^{\mathrm{T}}\sum_{e}\boldsymbol{G}^{\mathrm{T}}\left(\frac{1}{2}\int_{V_e}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}\mathrm{d}V\boldsymbol{G}\boldsymbol{a}-\int_{V_e}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{\varepsilon}_0\mathrm{d}V+\int_{V_e}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{\sigma}_0\mathrm{d}V-\int_{V_e}\boldsymbol{N}^{\mathrm{T}}\boldsymbol{f}\mathrm{d}V+\int_{S_\sigma^e}\boldsymbol{N}^{\mathrm{T}}\tilde{\boldsymbol{T}}\mathrm{d}S\right)$$

$$=\frac{1}{2}\boldsymbol{a}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{a}-\boldsymbol{a}^{\mathrm{T}}\boldsymbol{P}$$

总位能的变分$\delta\boldsymbol{\Pi}_{\mathrm{p}}=\dfrac{\partial\boldsymbol{\Pi}_{\mathrm{p}}}{\partial\boldsymbol{a}}=0$,得到有限元求解方程

$$\boldsymbol{K}\boldsymbol{a}=\boldsymbol{P}$$

其中

$$\boldsymbol{K}=\sum_{e}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{K}^{e}\boldsymbol{G}$$

则单元刚度矩阵的计算公式是

$$\boldsymbol{K}^e = \int_{V_e} \boldsymbol{B}^{\mathrm{T}} \boldsymbol{D} \boldsymbol{B} \mathrm{d}V$$

还要注意，上面Π_{P}表达式中的$\tilde{\boldsymbol{T}}$是作用在连续体边界上的力，包括两部分：作用在有关单元边界上的分布力($\boldsymbol{T}$)以及作用在节点上的集中力($\boldsymbol{F}$)。为了方便起见，将两部分分别表示如下

$$\boldsymbol{P}^e_{\varepsilon_0} = \int_{V_e} \boldsymbol{B}^{\mathrm{T}} \boldsymbol{D} \varepsilon_0 \mathrm{d}V;\ \boldsymbol{P}^e_{\mathrm{f}} = \int_{V_e} \boldsymbol{N}^{\mathrm{T}} \boldsymbol{f} \mathrm{d}V;\ \boldsymbol{P}^e_{\mathrm{S}} = \int_{S^e_\sigma} \boldsymbol{N}^{\mathrm{T}} \boldsymbol{T} \mathrm{d}S;\ \boldsymbol{P}^e_{\sigma_0} = -\int_{V_e} \boldsymbol{B}^{\mathrm{T}} \boldsymbol{\sigma}_0 \mathrm{d}V$$

从而，载荷列阵可表示为

$$\boldsymbol{P} = \sum_e \boldsymbol{G}^{\mathrm{T}} (\boldsymbol{P}^e_{\mathrm{F}} + \boldsymbol{P}^e_{\mathrm{S}} + \boldsymbol{P}^e_{\sigma_0} + \boldsymbol{P}^e_{\varepsilon_0}) + \boldsymbol{P}_{\mathrm{F}}$$

其中$\boldsymbol{P}_{\mathrm{F}}$为节点集中力列阵。

上面的①～③是假定位移模式，求解广义坐标，最后得到单元插值函数，这三步是广义坐标有限元的内容。④、⑤是利用变分原理建立有限元格式的一般方法，这里用的是最小位能原理，建立以位移为基本场变量的有限元求解方程，求解平衡问题。

有了方程后，为了求解方程，还需引入强制边界条件，然后根据方程的特点进行求解，以得到节点位移，有些还需要进行某些辅助计算。

6.7 小结

随着现代科学技术的发展，人们正在不断建造更为快速的交通工具、更大规模的建筑物、更大跨度的桥梁、更大功率的发电机组和更为精密的机械设备。这一切都要求工程师在设计阶段就能精确地预测出产品和工程的技术性能，需要对结构的静、动力强度以及温度场、流场、电磁场和渗流等技术参数进行分析计算。例如分析计算高层建筑和大跨度桥梁在地震时所受到的影响，看看是否会发生破坏性事故；分析计算核反应堆的温度场，确定传热和冷却系统是否合理；分析涡轮机叶片内的流体动力学参数，以提高其运转效率。这些都可归结为求解偏微分方程，显然解析计算往往是不现实的。

分布参数系统仿真技术为解决复杂的工程分析计算问题提供了有效的途径。经过多年的研究，分布参数系统仿真已经取得了很大进展，特别是20世纪60年代以来得到迅速发展的有限元方法，已经得到广泛的应用。典型的分布参数系统仿真软件有：

(1) LS-DYNA　这是LSTC公司的系列软件，由J. O. Hallquist主持于1976年在美国劳伦斯利弗莫尔国家实验室(Lawrence Livermore National Lab.)开发完成的，主要目的是为核武器的弹头设计提供分析工具，后经多次扩充和改进，计算功能更为强大。

(2) DYTRAN　这是MSC. Software Corporation(MSC公司)的产品。该软件在LS-DYNA3D的框架下，在程序中增加荷兰PISCES INTERNATIONAL公司开发的PICSES的高级流体动力学和流体——结构相互作用功能，还在PISCES的欧拉模式算法基础上，开发了物质流动算法和流固耦合算法。

(3) ABAQUS　这是HKS公司的产品，是一套先进的通用有限元系统，也是功能最强的有限元软件之一，可以分析复杂的固体力学和结构力学系统。ABAQUS有两个主要分

析模块：ABAQUS/Standard 提供通用的分析能力，如应力和变形、热交换、质量传递等；ABAQUS/Explicit 提供对时间进行显示积分求解。

(4) ANSYS　由美国 ANSYS 开发，是融结构、流体、电场、磁场、声场分析于一体的大型通用有限元分析软件。它能与多数 CAD 软件接口，实现数据的共享和交换，如 Pro/Engineer，NASTRAN，Alogor，I-DEAS，AutoCAD 等，是现代产品设计中的高级 CAD 工具之一。

(5) COSMOS　美国 SRAC 公司在 1993 年推出，该产品的特点是：传统有限元分析的数值计算方法之中，有直接计算法(direct solver)与迭代法(iterative)两种。过去，迭代法一直无法直接而有效地保证数值计算的收敛性，因此，直接计算法在多数有限元素分析软件中仍然是一种主流的计算方法。SRAC 公司 1993 年将保证收敛的迭代法——又称做快速有限元法导入 COSMOS 的产品之中。

(6) MSC. Nastran　1969 年 NASA 推出了其第一个 NASTRAN 版本，经过 30 多年的开发和改进，主要功能模块有：基本分析模块(含静力、模态、屈曲、热应力、流固耦合及数据库管理等)、动力学分析模块、热传导模块、非线性分析模块、设计灵敏度分析及优化模块、超单元分析模块、气动弹性分析模块、DMAP 用户开发工具模块及高级对称分析模块。

差分法及有限元法是分布参数系统仿真的两类基本方法。由于篇幅限制，本书仅对差分法和有限元法的原理进行简单介绍，主要目的是让读者对分布参数系统仿真建模的知识有一个较为全面的了解。若要深入掌握这些方法，还需要进一步学习专门的书籍。

习题

1. 讨论对流方程 $\frac{\partial u}{\partial t}+a\frac{\partial u}{\partial x}=0, a>0$ 的差分格式 $\frac{u_j^{n+1}-u_j^n}{\tau}+a\frac{u_j^{n+1}-u_{j-1}^{n+1}}{h}=0$ 的截断误差及稳定性。

2. 讨论扩散方程 $\frac{\partial u}{\partial t}-b\frac{\partial^2 u}{\partial x^2}=0, b>0$ 的差分格式 $\frac{3}{2}\frac{u_j^{n+1}-u_j^n}{\tau}-\frac{1}{2}\frac{u_j^n-u_j^{n-1}}{\tau}-b\frac{u_{j+1}^{n+1}-2u_j^n+u_{j-1}^{n+1}}{h^2}=0$ 的截断误差及稳定性。

3. 用五点差分格式的一般迭代法求解 Poisson 方程的解。Poisson 方程的形式为

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=16, & (x,y)\in D\\ u=0, & (x,y)\in \partial D\end{cases}$$

试列出差分方程(其中有界区域 $D=\{(x,y)\,|\,|x|,|y|<1\}$，$\partial D$ 由四条直线段组成：$\partial D=\{(x,y)\,|\,x=0,1,0\leqslant y\leqslant 1;\ y=0,1,0\leqslant x\leqslant 1\}$)，并对 $h=1,1/2$ 分别求解。

4. 对上题用松弛法求解，取 $\omega=1.5$，扫描顺序是先按行，在一行内按列。比较计算结果。

5. 一维热传导的微分方程如下

$$A(\phi)=\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2}+Q=0 \quad (0\leqslant x\leqslant L)$$

其中

$$Q(x)=\begin{cases}1, & \text{当 } 0\leqslant x\leqslant L/2\\ 0, & \text{当 } L/2<x\leqslant L\end{cases}$$

边界条件是：在 $x=0$ 和 $x=L$ 时，$\phi=0$。若取傅里叶级数作为近似解

$$\phi\approx\tilde{\phi}=\sum_{i=1}^{n}a_i\sin\frac{i\pi x}{L}$$

其中 a_i 为待定参数，试探函数为 $N_i=\sin\frac{i\pi x}{L}$。试用加权余量的配点法、子域法和伽辽金法，近似取一项及两项求解，比较其结果。

参考文献

[1] 冯康等. 数值计算方法. 北京：国防工业出版社，1978
[2] 陆金甫，关治. 偏微分方程数值解法. 北京：清华大学出版社，1989
[3] 熊光楞. 数字仿真算法与软件. 北京：宇航出版社，1991
[4] Spriet J A 等著. 王正中等译. 计算机辅助建模与仿真. 北京：科学出版社，1991
[5] 王勖成，邵敏. 有限单元法基本原理和数值方法. 第2版. 北京：清华大学出版社，1995
[6] 胡建伟，汤怀民. 微分方程数值方法. 第2版. 北京：科学出版社，2006

第二篇　离散事件系统仿真篇

第 7 章
离散事件系统仿真基础

7.1 基本概念

在前面各章中讨论的系统，其状态变量是连续变化的(时间上可以是连续的，也可以是离散的)，这类系统的仿真称为连续系统仿真。

从本章开始，我们将讨论另一类性质完全不同的系统，这就是离散事件系统。这类系统中的状态只是在离散时间点上发生变化，而且这些离散时间点一般是不确定的。

例 7.1 单人理发馆系统，设上午 9:00 开门，下午 5:00 关门，顾客的到达时间一般是随机的，为每个顾客服务的时间长度也是随机的。描述该系统的是服务台的状态(忙或闲)、顾客排队等待的队长。显然，这些状态量的变化也只能在离散的随机时间点上发生。

■

类似的例子还很多，如订票系统、库存系统、加工制造系统、交通系统、计算机系统、网络系统等。

由于离散事件系统固有的随机性，对这类系统的研究往往十分困难。经典的概率及数理统计理论、随机过程理论虽然为研究这类系统提供了理论基础，并能对一些简单系统提供解析解，但对工程实际中的大量系统，唯有依靠计算机仿真技术才能提供较为完整的结果。

下面首先介绍在离散事件系统仿真中所用到的一些基本概念。

1. 实体

实体是描述系统的三个基本要素之一。在离散事件系统中的实体可分为两大类：临时实体及永久实体。在系统中只存在一段时间的实体叫临时实体。这类实体由系统外部到达系统，通过系统，最终离开系统。如例 7.1 中的顾客显然是临时实体，它按一定规律到达，经过服务员服务(可能要排队等待一段时间)后即离开系统。那些虽然到达但未进入理发馆的

顾客则不能称为该系统的临时实体。永久驻留在系统中的实体称为永久实体。如例7.1中的服务员就是永久实体。只要系统处于活动状态，这些实体就存在，或者说，永久实体是系统处于活动状态的必要条件。

临时实体按一定规律不断地到达(产生)，在永久实体作用下通过系统，最后离开系统，整个系统呈现出动态过程。

2. 事件

描述离散事件系统的另一个重要概念就是事件。事件是引起系统状态发生变化的行为。从某种意义上说，这类系统是由事件来驱动的。

例如，在例7.1中，可以定义“顾客到达”为一类事件，由于顾客到达，系统的状态——服务员的“状态”可能从闲变到忙(如果无人排队)，或者另一系统状态——排队的顾客人数发生变化(队列人数加1)。一个顾客接受服务完毕后离开系统也可以定义为一类事作，因为服务台“状态”由忙变成闲。

在一个系统中，往往有许多类事件，而事件的发生一般与某一类实体相联系，某一类事件的发生还可能会引起别的事件发生，或者是另一类事件发生的条件等。为了实现对系统中的事件进行管理，仿真模型中必须建立事件表，表中记录每一个发生了的或将要发生的事件类型、发生时间以及与该事件相联的实体的有关属性等。

在仿真模型中，由于是依靠事件来驱动，除了系统中固有事件(又称为系统事件)外，还有所谓的“程序事件”，它用于控制仿真进程。例如，如果要对例7.1的系统进行从上午9:00开门到下午5:00关门这一段时间内的动态过程仿真，则可以定义“仿真时间达到8小时后停止仿真”作为一个程序事件，当该事件发生时即结束仿真模型的执行。

3. 活动

离散事件系统中的活动，通常用于表示两个可以区分的事件之间的过程，它标志着系统状态的转移。例7.1中，顾客的到达事件与该顾客开始接受服务事件之间可称为一个活动，该活动使系统的状态(队长)发生变化，顾客开始接受服务到服务完毕后该顾客离去也可视为一个活动，它使队长减1或/和使服务台由忙变成闲。

4. 进程

进程由若干个有序事件及若干有序活动组成。一个进程描述了它所包括的事件及活动间的相互逻辑关系及时序关系。如例7.1中，一个顾客到达系统，经过排队、接受服务，直到服务完毕后离去可称为一个进程。

事件、活动、进程三者之间的关系可用图7.1描述。

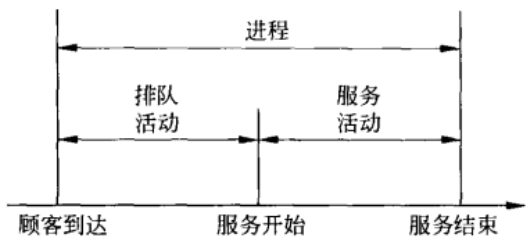


图7.1 事件、活动、进程三者关系示意图

5. 仿真钟

仿真钟表示仿真时间的变化。在连续系统仿真中，将连续模型进行离散化而成为仿真模型时，仿真时间的变化基于仿真步长的确定，可以是定步长，也可以是变步长。离散事件动态系统的状态本来就只在离散时间点上发生变化，因而不需要进行离散化处理。但是，由于引起状态变化的事件发生时间的随机性，仿真钟的推进步长则完全是随机的；而且，两个相邻发生的事件之间系统状态不会发生任何变化，因而仿真钟可以跨过“不活动”的周期。从一个事件发生时刻推进到下一事件发生时刻，仿真钟的推进呈现跳跃性，推进速度具有随机性。可见，仿真模型中时间控制部件是必不可少的，以便按一定规律来控制仿真钟的推进。

6. 统计计数器

连续系统仿真的目的是要得到状态变量的动态变化过程并由此分析系统的性能。离散事件系统的状态随着事件的不断发生也呈现出动态变化过程，但仿真的目的主要不是要得到这些状态是如何变化的；因为这种变化是随机的，某一次仿真运行得到的状态变化过程只不过是随机过程的一次取样。如果进行另一次独立的仿真运行所得到的状态变化过程可能完全是另一种情况。它们只有在统计意义下才有参考价值。

例如，例 7.1 中，由于顾客到达的时间间隔具有随机性，服务员为每一个顾客服务的时间长度也是随机的，因而在某一时刻，顾客排队的队长或服务台的忙闲情况完全是不确定的。从分析该系统的角度来看，感兴趣的可能是系统的平均队长，顾客的平均等待时间，或者是服务员的利用率等。在仿真模型中，需要有一个统计计数部件，以便统计系统中的有关变量。

从上述的讨论中可以看到，离散事件系统与连续系统有本质的区别，因而对这类系统的仿真所采用的方法以及所研究的内容也是很不相同的。

7.2 仿真钟的推进

上一节简要地介绍了仿真钟的基本概念，本节就离散事件系统仿真中的仿真钟推进方法进行详细的讨论，因为这是离散事件系统仿真的基础。

离散事件系统仿真的仿真钟推进方法有两大类：第一类是按下一最早发生事件的发生时间推进，亦称为事件调度法；另一类是固定增量推进方法。先讨论第一类方法。

如前一节所述，仿真模型中的时间控制部件用于控制仿真钟的推进。事件调度法中，事件表按事件发生时间先后顺序安排事件。时间控制部件从事件表中始终选择具有最早发生时间的事件记录，然后将仿真钟修改到该事件发生时刻。对每一类事件，仿真模型有相应的事件子程序。每一个事件记录包含该事件的若干个属性，其中事件类型是必不可少的，要根据事件类型调用相应的事件子程序。在事件子程序中，处理该事件发生时系统状态的变化，进行用户所需要的统计计算；如果是条件事件，则应首先进行条件测试，以确定该事件是否确能发生；如果条件不满足，则推迟或取消该事件。该事件子程序处理完后返回时间控制

部件。这样，事件的选择与处理不断地进行，仿真钟不断地从一个事件发生时间推进到下一最早发生的事件发生时间，直到终止仿真的条件或程序事件发生时停止仿真。

下面通过一个具体的例子来加以说明，以便读者更进一步地理解事件调度法。

例 7.2 单服务台排队系统，如单窗口的售票站。设：

$A_i = t_i - t_{i-1}$ 为第 $i-1$ 个与第 i 个顾客到达之间的间隔时间；

S_i = 服务员为第 i 个顾客服务的时间长度；

D_i = 第 i 个顾客排队等待的时间长度；

$C_i = t_i + D_i + S_i$ 为第 i 个顾客离去的时间；

t_i = 第 i 个任何一类事件发生的时间；

b_i = 第 i 个任何一类事件发生的时间；

q_i = 第 i 个事件发生时的队长；

Z_i = 第 i 个事件发生时服务员的状态，其中 $Z_i = 1$ 表示忙，$Z_i = 0$ 表示闲。

若定义如下系统事件类型：

类型 1 顾客到达事件，类型 2 顾客接受服务事件，类型 3 顾客接受服务完毕并离去的事件。

定义程序事件为：

仿真运行到 150 个时间单位(例如分钟)结束。

该系统的模型可用如图 7.2 的流程来描述。

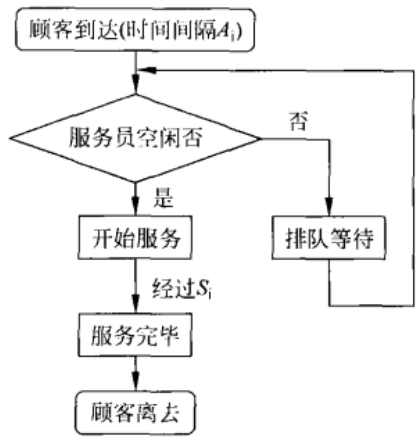


图 7.2 单服务台售票站仿真模型

一般说来，A_i，S_i 是随机变量，要根据其分布函数来产生，为了便于解释，假定我们已经得到了这些随机变量的样本值为

$$A_1 = 15,\quad A_2 = 32,\quad A_3 = 24,\quad A_4 = 40,\quad A_5 = 22,\cdots$$

$$S_1 = 43,\quad S_2 = 36,\quad S_3 = 34,\quad S_4 = 28,\cdots$$

为了加深读者对事件表这一离散事件系统仿真核心问题的理解，表 7.1 列出了单服务台售票站的事件表。

表 7.1 单服务台售票站的事件表

时间	事 件	服务员状态	排队长度
0	仿真开始	0	0
15	顾客 1 到达	1	0
47	顾客 2 到达	1	1
58	顾客 1 服务完毕	0	1
58	顾客 2 接受服务	1	0
71	顾客 3 到达	1	1
94	顾客 2 服务完毕	0	1
94	顾客 3 接受服务	1	0
…	…	…	…
150	仿真结束		

结合表 7.1 列出的事件表，该模型的下一事件推进法的仿真钟推进过程如图 7.3。

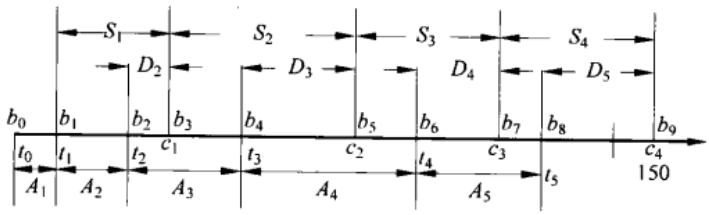


图 7.3 事件调度法仿真钟推进

下面具体分析仿真钟的推进过程。

系统初始状态 $q_0=0, Z_0=0$。

仿真钟初始值 $TIME=b_0=t_0$。

在 $TIME=t_0$ 时刻，下一最早发生事件为第 1 号顾客到达，发生时刻为 b_1，即 $t_1=b_1$。

$$t_1 = t_0 + A_1 = 15$$

因 $t_1<150$。仿真钟推进到 t_1，然后处理该事件。由于是到达事件，且 $Z_0=0$，立即为刚刚到达的顾客服务，即 $D_1=0$，该顾客的服务时间为 $S_1=43$，则其离去时间为

$$C_1 = t_1 + S_1 + D_1 = 15 + 43 + 0 = 58$$

此时服务台状态由 $Z_0=0$ 变为 $Z_0=1$。

下一最早发生事件仍是到达事件，因为

$$t_2 = t_1 + A_2 = 15 + 32 = 47 < C_1$$

从而 $b_2=t_2=47$。

仿真钟推进到 t_2。处理该到达事件，因 $Z_1=1$，该到达顾客只好排队等待，从而队长 $q_2=q_1+1=1$，该顾客开始等待时间为 t_2。

下一最早发生事件应是第 1 号顾客离去事件，因为下一到达事件发生的时间 t_3 为

$$t_3 = t_2 + A_3 = 47 + 24 = 71 > C_1$$

从而 $b_3=C_1$。

仿真钟推进到 $C_1=58$，处理第 1 号顾客的离去事件，包括：统计服务人数，观察队列中是否有顾客等待，目前 $q_2=1$，则该顾客进入服务，同时要计算其排队等待的时间 $D_2=C_1-t_2$，队长 $q_3=q_2-1=0$，该顾客的服务时间为 $S_2=36$，其离去时间 $C_2=C_1+S_2=94$。

由 $C_2>t_3$，因而下一事件应是到达事件，仿真钟推进到 $b_4=t_3$，…。

依次下去，直到下一事件为仿真结束的程序事件为止。

■

除了按下一最早发生时间的事件推进仿真钟的方法之外，早期的离散事件系统仿真也采用过另一种推进方法，即固定增量时间推进。这种方法类似于连续系统仿真中的等步长策略，现简要说明如下。

选择适当的时间单位 T 作为仿真钟推进时的增量，每推进一步进行如下处理：

(1) 该步内若无事件发生，则仿真钟再推进一个单位时间 T。

(2) 若在该步内有若干个事件发生，则认为这些事件均发生在该步的结束时刻。为便于进行各类事件处理，用户必须规定当出现这种情况时各类事件处理的优先顺序。

这种方法的缺点是显而易见的。首先，仿真钟每推进一步，均要检查事件表以确定是否

有事件发生，增加了执行时间；其次，该步任何事件的发生均认为发生在这一步的结束时刻，如果 T 选择过大，则会引入较大的误差；而且要求用户确定各类事件的处理顺序，增加了建模的复杂性。

固定增量推进法主要用于系统事件发生时间具有较强周期性的模型，如定期订货的库存系统，以年、月为单位的经济计划系统等。

对于大多数离散事件系统来说，由于一般都具有较强的随机性，本书中采用事件调度法来讨论离散事件系统仿真的有关内容。

本章中，我们先讨论两类最常见的离散事件系统——排队系统及库存系统的仿真，以便使读者对离散事件系统仿真有一个初步的理解。

7.3 单服务台排队系统仿真

7.3.1 排队系统的基本概念

1918年，Erlang提出排队论，并将它用于电话系统，其实质就是研究服务台与顾客之间的效率问题，希望服务台效率高，而顾客的等待时间又不太长，它又称为随机服务理论。

在排队系统中，下列概念是经常遇到的。

(1) 实体(顾客)到达模式

实体到达模式一般用到达时间间隔来描述，可分为确定性到达及随机性到达。随机性到达采用概率分布来描述，最常采用的是泊松到达。

平稳泊松过程可描述为：在$(t,t+s)$内到达的实体数 k 的概率为

$$P\{N(t+s)-N(t)=k\}=\frac{\mathrm{e}^{-\lambda s}(\lambda s)^k}{k!} \tag{7.1}$$

其中，$N(t)$表示在$(0,t)$区间内到达实体的个数；$t\geqslant 0$；$s\geqslant 0$；$k=0,1,2,\cdots$；λ 为到达速率。

若实体到达满足平稳泊松过程，则到达时间间隔服从指数分布，其密度函数为

$$f(t)=\lambda \mathrm{e}^{-\lambda t}=\frac{1}{\beta}\mathrm{e}^{-t/\beta} \quad (t\geqslant 0) \tag{7.2}$$

其中，$\beta=1/\lambda$ 为到达时间间隔均值。

(2) 服务模式

服务台为顾客服务的时间可以是确定性的，也可能是随机的，后者采用服务时间的概率分布来描述。

(3) 排队规则

表示服务台完成当前的服务后，从队列中选择下一实体的原则，一般有：

FIFO——先到先服务；

LIFO——后到先服务；

按优先级别服务——根据队列中实体的重要程度选择最优先服务者。

(4) 服务流程

当系统中有多个服务台，有多个队列时，服务台如何从某一个队列中选择某一个实体服

务则称为服务流程问题，它包括各队列之间的关系，如实体可否换队以及换队规则等。

排队系统中的上述四个特征，一般用符号 GI/G/S 来表示，其中：

GI(general independent)表示到达模式，若为平稳泊松过程，其到达时间间隔服从指数分布，用 M 表示(马尔可夫过程)；若为 Erlang 分布，则用 E_K 表示，K 表示 Erlang 分布的维数；若是确定性时间间隔，则用 D 表示。

G(general)表示服务时间的分布，分布函数的符号与 GI 相同。

S 表示单队多服务台的数目，且按 FIFO 规则服务。

例如，一个具有指数分布的到达时间间隔，服务时间也服从指数分布，且按 FIFO 规则服务的单服务台单队排队系统可记为 M/M/1。

研究排队系统的目的是得到系统的统计性能，比较普遍使用的性能指标有以下四种：

(1) 稳态平均延误时间 d

$$d = \lim_{n\to\infty} \sum_{i=1}^{n} D_i / n \tag{7.3}$$

其中，D_i 为第 i 个实体的延误时间，n 是接受服务的实体数。平均延误时间就是实体在队列中的平均等待时间。

(2) 实体通过系统的稳态平均滞留时间 w

$$w = \lim_{n\to\infty} \sum_{i=1}^{n} W_i / n = \lim_{n\to\infty} \sum_{i=1}^{n} (D_i + S_i) / n \tag{7.4}$$

其中，W_i 为第 i 个实体通过系统时的滞留时间，它等于实体在队列中的等待时间 D_i 与该实体接受服务的时间 S_i 之和。

(3) 稳态平均队长 Q

$$Q = \lim_{T\to\infty} \int_0^T Q(t)\mathrm{d}t / T \tag{7.5}$$

其中，$Q(t)$为 t 时刻的队列长度，T 为系统运行时间。

(4) 系统中稳态平均实体数 L

$$L = \lim_{T\to\infty} \int_0^T L(t)\mathrm{d}t / T = \lim_{T\to\infty} \int_0^T (Q(t) + S(t))\mathrm{d}t / T \tag{7.6}$$

其中，$L(t)$为 t 时刻系统中的实体数，它是在队列中的实体数 $Q(t)$与正在接受服务的实体数 $S(t)$之和。

上述四个性能指标存在的条件是服务台的利用率 $\rho<1$，ρ 的定义是

$$\rho = \frac{\text{平均服务时间}}{\text{平均到达时间间隔}} \tag{7.7}$$

对 M/M/1 排队系统，上述四项指标可解析计算得到，即

$$d = Q/\lambda, \quad w = L/\lambda \tag{7.8}$$

$$Q = \frac{\rho^2}{1-\rho}, \quad L = \frac{\rho}{1-\rho} \tag{7.9}$$

7.3.2 单服务台排队系统仿真举例

本节以例 7.1 为例，讨论事件调度法实现离散事件系统仿真的有关内容，先将系统中的

有关参数说明如下：

(1) 实体到达模式

该系统中的临时实体是顾客，实体到达模式就是顾客到达模式，设到达时间间隔 A_i 服从均值为 $\beta_A=5\text{min}$ 的指数分布，即

$$f(A)=\frac{1}{\beta_A}e^{-A/\beta_A} \quad (A\geqslant 0) \tag{7.10}$$

(2) 服务模式

设服务员为每个顾客服务的时间为 S_i，它也服从指数分布，均值为 $\beta_S=4\text{min}$，即

$$f(S)=\frac{1}{\beta_S}e^{-S/\beta_S} \quad (S\geqslant 0) \tag{7.11}$$

(3) 服务规则

由于是单服务台系统，考虑系统中顾客按单队排队，按 FIFO(先入先出)方式服务，由上述可知，该系统是 M/M/1 系统。

在进行该系统仿真时，首先必须产生具有给定分布的随机变量。例如，为了确定第 $i-1$ 个与第 i 个顾客到达之间的时间间隔 A_i，要从式(7.10)中产生。又如为了确定服务员为第 j 个顾客的服务时间 S_j，则必须从式(7.11)中产生，关于如何从某一分布中产生随机变量的一般方法将在第 8 章中讨论。在本例这种简单的情况下，可以采用 $f(x)$对应的分布函数 $F(x)$反函数的方法(亦称为反变换法)，现说明如下：

令 u 是取值为[0,1]范围内服从均匀分布的随机变量，即

$$u=F(x)=\begin{cases}0, & x<0\\ x, & 0\leqslant x\leqslant 1\\ 1, & x>1\end{cases} \tag{7.12}$$

反变换法要求用 u 对 $F(A)$或 $F(S)$进行取样，即令

$$u_1=F(A)=1-e^{-A/\beta_A}$$

则

$$A=-\beta_A\ln(1-u_1)$$

由于 u_1 为[0,1]之间均匀分布的随机变量，则 $1-u_1$ 也是[0,1]之间均匀分布的随机变量，因而可令

$$A=-\beta_A\ln u_1 \tag{7.13}$$

同样，对 $F(S)=1-e^{-S/\beta_S}$，也可得到

$$S=-\beta_S\ln u_2 \tag{7.14}$$

事件调度法要求对引起系统状态发生变化的系统事件加以定义，并建立相应的事件表。

定义系统事件是仿真建模十分重要的阶段。首先要根据仿真的目的及系统的内部运动规律确定系统的状态变量；不同的系统，其系统状态定义当然不同，即使是同一系统，仿真的目的不同，系统状态定义也可能不尽相同。然后在定义系统状态的基础上定义系统事件及其有关的属性。一般来说，在事件调度法中，事件的类型及其发生时间是必不可少的事件属性，当然还可能有其他的属性。

本系统比较简单，系统的状态可以采用顾客排队的队长 q 及服务员忙闲状态 Z 来描述，相应地，引起这些状态发生变化的事件有：顾客到达系统事件，顾客接受服务事件，顾客接受完服务后离开系统事件。这三类事件的类型及属性见表 7.2。

表 7.2 单服务台三类事件的类型与属性

事件类型	事件描述	属性
第 1 类	顾客到达系统	到达时间
第 2 类	顾客接受服务	开始服务时间
第 3 类	顾客接受完服务后离开系统	离去时间

事件类型及其有关属性值均记录在事件表中。

仿真的目的在于分析系统的性能，因而需要有一个统计计数器来搜集仿真过程中的有关数据并进行统计分析，采用哪些性能来度量系统以及如何在仿真过程中得到这些性能也是仿真中特别需要加以考虑的问题。

对于例 7.1，假设仿真的目的是要求估计服务 n 个顾客后的顾客平均队长 $Q(n)$ 及平均排队等待时间 $d(n)$，这可以参照式(7.3)及式(7.5)得到相应的计算式

$$d(n) = \overline{D}(n) = \sum_{i=1}^{n} D_i / n \tag{7.15}$$

$$Q(n) = \overline{Q}(n) = \frac{1}{T}\int_0^T Q(t)\,\mathrm{d}t \tag{7.16}$$

其中，D_i 为第 i 个顾客排队等待时间，$Q(t)$ 为 t 时刻排队等待的顾客数，T 为完成 n 个顾客服务所耗时间，$d(n)$，$Q(n)$ 表示估计值，$\overline{D}(n)$，$\overline{Q}(n)$ 表示平均值。

由于事件调度法的仿真钟推进的随机性，为计算方便，式(7.16)计算 $Q(n)$ 采用式(7.17)的形式，即

$$Q(n) = \frac{1}{T}\sum_{i=1}^{n} R_i \tag{7.17}$$

式中，R_i = 在时间区间$[b_{i-1}, b_i]$上排队人数 q_i 乘以该时间区间长度$(b_i - b_{i-1})$，即

$$R_i = q_i(b_i - b_{i-1})$$

其中 b_i 是第 i 个任何一类事件发生的时间；

n = 在区间$[t_0, t_0 + T]$上发生的事件总数，其中 t_0 为仿真初始时间。

用式(7.17)代替式(7.16)的原因在于，在任何相邻的事件时间区间上，系统的状态不发生变化，因而顾客排队的队长 q_i 在$(b_i - b_{i-1})$内保持常数，那么在对 $Q(t)$ 进行积分计算时只需计算 R_i 就可以了。在$[t_0, t_0 + T]$上发生的事件总数为 n，将该 n 个 R_i 累加起来后再除以 T，显然就是平均队长，采用式(7.17)来计算 $Q(n)$ 从程序实现的角度来看则更易于处理。式(7.16)与式(7.17)的等价性可由图 7.4 来表示。

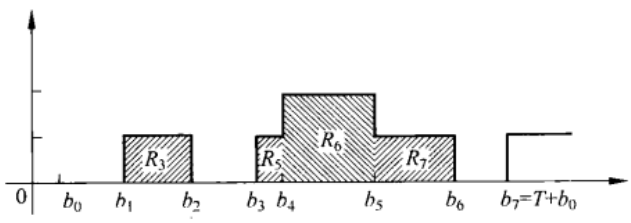


图 7.4 $\int_0^T Q(t)\,\mathrm{d}t$ 与 $\sum_{i=1}^{n} R_i$ 等价的图示

在该图中，b_1，b_2，b_4，b_5 时刻发生顾客到达，在 b_3，b_6，b_7 时刻发生顾客离去。

对例 7.1 做了上述说明以后，就可以编制该系统的仿真程序了。按事件调度法推进仿真钟的机理，其仿真程序的结构可用图 7.5 来表示。

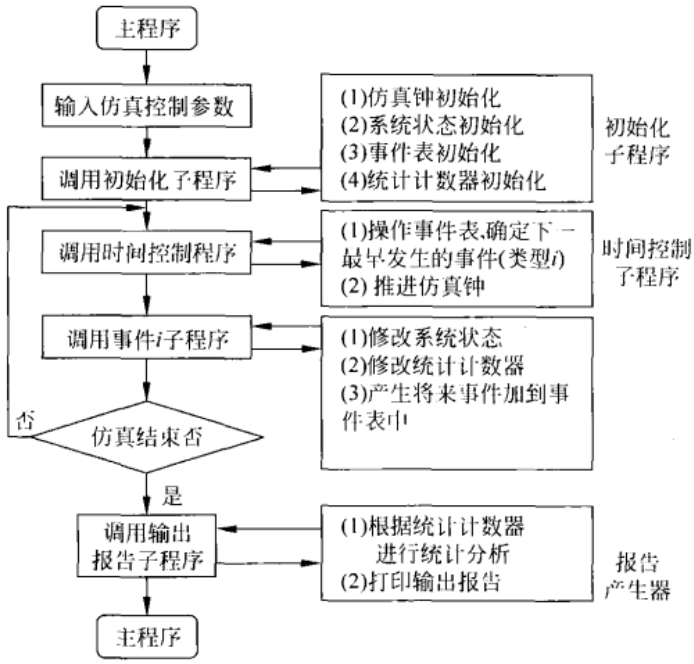


图 7.5 事件调度法程序结构

这里，我们给出 M/M/1 排队系统的一个仿真实例的结果。设顾客的平均到达时间间隔为 5 分钟，服从指数分布，为每个顾客服务的时间也服从指数分布，均值为 4 分钟，仿真运行长度为到服务完 3 000 个顾客结束，下面是按上述仿真模型运行的输出结果。

```
======Single Server Queuing System======
Mean Interarrival Time:          5.0 Minutes
Mean Service Time:               4.0 Minutes
Number of Simulated Customers    3000
Average Delay in Queue           15.563 Minutes
Average Number in Queue          3.181
```

从上面的仿真结果报告中可以看到，得到的平均队长为 3.181 人，平均每个顾客的等待时间为 15.563 分钟。仿真的结果是否正确呢？对 M/M/1 系统，可以解析得到上述系统性能的稳态理论值，它们是

由 $\beta_A=5.0\text{min}$，$\beta_S=4\text{min}$，可得 $\lambda=0.2$，$\rho=0.8$，因而

$$Q=\frac{\rho^2}{1-\rho}=3.2 \quad d=Q/\lambda=16.0$$

可见，$\bar{Q}(3\,000)=3.181$，$\bar{d}(3\,000)=15.563$ 是很接近稳态理论值的，因而有理由认为仿真的结果是可信的。

但是，需要注意的是，并不是在任何情况下都能得到这样准确的仿真结果，实际上，若分别仿真运行服务完 $n=1\,000$、2 000、3 000、5 000 个顾客结束，其仿真结果见表 7.3。

表 7.3 仿真运行服务完 $n=1\,000,2\,000,3\,000,5\,000$ 个顾客的结果

仿真长度	1 000	2 000	3 000	5 000
$\bar{Q}$	3.916	3.62	3.181	3.425
$\bar{d}$	19.723	17.586	15.563	16.982

产生这种现象的根本原因在于离散事件系统的随机性。模型的随机性决定了系统性能取值的随机性。由于每次仿真运行的结果只是对表征系统性能的随机变量的一次取样。那么，当系统比较复杂时，如何对仿真结果的可信性进行判断，是离散事件系统仿真中十分重要的内容。关于仿真结果分析的有关内容，将在第 10 章中进行讨论。

7.4 库存系统的仿真

离散事件系统仿真研究的另一大类系统是库存系统，它不仅包括一般意义下的库存系统，如商品、器材的管理，银行现金的管理，水库库存水量的管理等，还包括人才储备及管理这样广义的库存系统。

7.4.1 库存系统的基本概念

正如在 7.3 节中看到的那样，在排队系统中，由于顾客不断地按一定规律到达，服务台按一定规则为到达系统的顾客服务，然后顾客离开系统，这样系统的状态发生动态变化。

在库存系统中，库存量的变化是由以下两个方面的因素引起的。首先是顾客的需求，由于满足了需求，库存量不断减少，为保证供应，就需要补充库存，称为订货。如同排队系统中的“到达”与“服务”一样，“需求”与“订货”是库存系统的两个最基本的概念。由于需求与订货的不断发生，库存量呈现动态变化。

根据需求与订货的规律，可将库存系统分为两大类。一类是确定性库存系统，另一类则是随机库存系统。

在确定性库存系统中，需求量是确定性的，需求发生时间也是确定性的，同时，订货与订货发生时间是确定性的，且从订货到货物入库的时间都是确定性的，对这种库存系统，可以解析地进行研究。

随机库存系统则比确定性库存系统要复杂得多。首先，需求发生时间可能是随机的，每次需求量也可能是随机的；另外，订货时间及订货量也可能是随机的，即使它们是确定性的，从订货到货物入库的时间也可能是随机的。对随机库存系统，一般只有通过仿真才有可能进行较深入的研究。

研究库存系统的目的一般是要确定或比较各种库存策略，它包括在不同的需求情况下，何时订货，订多少货为宜等。

库存策略的优劣一般采用“费用”高低来衡量，最常考虑的费用包括以下几个方面：

(1) 保管费：包括仓库、设备、人力、货物保存、损坏变质等支出费用，一般可折算成每

件每日费用或每件每月费用等。

(2) 订货费：包括货物本身的费用、订货手续费、运输费等。

(3) 缺货损失费：由于货物不足造成供不应求，错过销售机会或停工待料等造成的损失。

下面通过一个简单的例子，讨论如何对库存系统进行离散事件仿真。

7.4.2 确定性库存系统

设定库存的初始水平 Q_0，单位时间的需求量是常值，年订货量 D 为常值。如果认为订货无滞后，即一旦订货，所订货物即到达。采用安全库存策略，即不出现缺货，订货的策略是：用完后即重新订货，每次订货量 Q 与初始水平 Q_0 相同，则这是库存系统中最简单的情况，可以用如下图形来描述(参见图 7.6)。

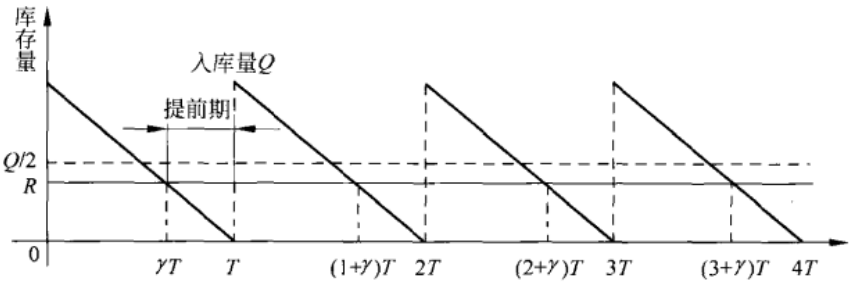


图 7.6 最简单的确定性库存系统

在图 7.6 中，T 为订货周期，即 $T=12/N$，N 为订货次数，即 $N=D/Q$。订货点发生在订货周期的结束时刻，无须考虑提前期。

设每次订货费用为 C_0，每件货物的保管费为 C_1，则不难计算出总费用 C 为

$$C = \frac{Q}{2}C_1 + \frac{D}{Q}C_0 \tag{7.18}$$

按总费用最小原则，$dC/dQ=0$，由式(7.18)可计算得到最佳订货量 Q^* 为

$$Q^* = \sqrt{2DC_0/C_1} \tag{7.19}$$

相应地，最小总费用 C^* 为

$$C^* = \sqrt{2DC_0C_1} \tag{7.20}$$

如果考虑订货滞后，对确定性货物到达滞后时间的库存系统来说，仍可采用上述模型，只需要确定订货的提前期 $(1-\gamma)T$。在考虑提前期的情况下，订货发生的时刻为 γT，此时的库存水平用 R 来表示，称为订货点。在这两种情况下，平均库存量均为 $0.5Q$。

上面讨论的是采用安全库存订货策略。如果在某一提前期下所订货物不能在 $nT(n=1,2,\cdots)$ 时刻到达，则可能出现两种情况，一是发生缺货，二是库存加大。若允许缺货，设一周期内缺货时间的百分比为 α，则此种情况下(考虑了提前期)的库存模型可用图 7.7 表示。由图不难计算出平均缺货量 Q_q，平均库存量 $\bar{Q}$ 及总费用 C 如下

$$Q_q = (1-\alpha)Q \cdot (1-\alpha)T/2T = 0.5(1-\alpha)^2Q \tag{7.21}$$

$$\bar{Q} = \alpha Q \cdot \alpha T/(2T) = 0.5\alpha^2 Q \tag{7.22}$$

$$C = C_0 \frac{D}{Q} + C_1 \cdot \frac{1}{2}\alpha^2 Q + C_2 \cdot \frac{1}{2}(1-\alpha)^2 Q \tag{7.23}$$

其中，C_2 表示每件缺货引起的损失费。按最小总费用原则，首先确定订货时间

$$\frac{\partial C}{\partial \alpha} = 0, \quad \alpha^* = \frac{C_2}{C_1 + C_2}$$

在 $\alpha^* T$ 点订货，确定最优订货量，即

$$\frac{\partial C}{\partial Q} = 0 \quad Q = \sqrt{\frac{2DC_0}{C_1}}\sqrt{\frac{C_1 + C_2}{C_2}} = \sqrt{\frac{2DC_0}{C_1}}\sqrt{\frac{1}{\alpha}} > \sqrt{\frac{2DC_0}{C_1}} \tag{7.24}$$

从而，最小总费用为

$$C = \sqrt{2DC_0C_1}\sqrt{\frac{C_2}{C_1 + C_2}} < \sqrt{2DC_0C_1} \tag{7.25}$$

可见，在允许缺货时的最小总费用比不允许缺货时的最小总费用小，而每次订货量要加大。

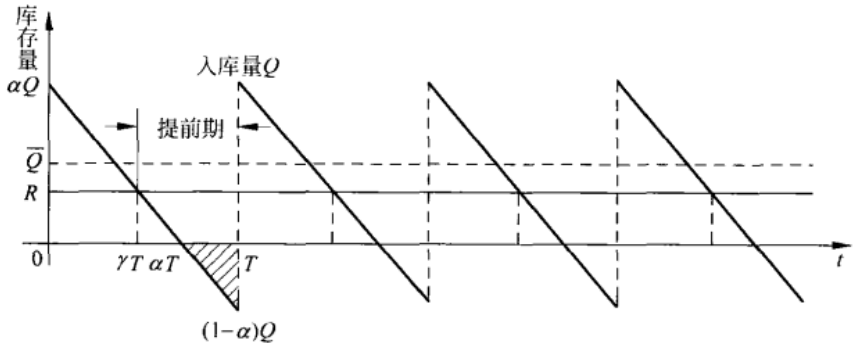


图 7.7 允许缺货的确定性库存系统

7.4.3 随机库存系统

一般的库存系统都具有随机性，即提前期 $(1-\gamma)T$ 是随机的，单位时间的需求量也是随机的。

先考虑最简单的随机库存系统，即每次订货量 Q 不变，订货点 R 不变，图 7.8 是这类系统的一种描述。现要求总费用最小的最优订货点、每次最优订货量及总费用。

设 I 为每周期期望库存量，S 为每周期期望缺货数，则

$$I = R + \frac{Q}{2} - E(y) \tag{7.26}$$

其中，$R+0.5Q$ 表示无提前期时每周期的期望库存量，y 为提前期内的需求量，是一随机变量，$E(y)$ 是 y 的期望值，而

$$S = \sum_{y=R}^{\infty}(y-R)h(y) \tag{7.27}$$

其中，$h(y)$ 是提前期需求量 y 的概率密度函数。每年所需费用为 C，则

$$C = C_0 \frac{D}{Q} + C_1 I + C_2 \frac{D}{Q} S \tag{7.28}$$

将式(7.26)、式(7.27)表达式代入式(7.28)，可得

$$C = C_0 \frac{D}{Q} + C_1\left(R + \frac{Q}{2} - E(y)\right) + C_2 \frac{D}{Q}\left[\sum_{y=R}^{\infty}(y-R)h(y)\right] \tag{7.29}$$

为得到最小总费用，先求最佳订货点 R 的值，即令 $\frac{\partial C}{\partial R}=0$，得 $\sum_{y=R}^{\infty} h(y)=\frac{C_0 Q}{C_2 D}$，也就是 $P(Y>R)=\frac{C_0 Q}{C_2 D}$。选择 R，使随机变量 $Y>R$ 的概率为 $\frac{C_0 Q}{C_2 D}$。

在此基础上求最小总费用，即令 $\frac{\partial C}{\partial Q}=0$，可得

$$Q = \sqrt{\frac{2D(C_0 + C_2 S)}{C_1}} \tag{7.30}$$

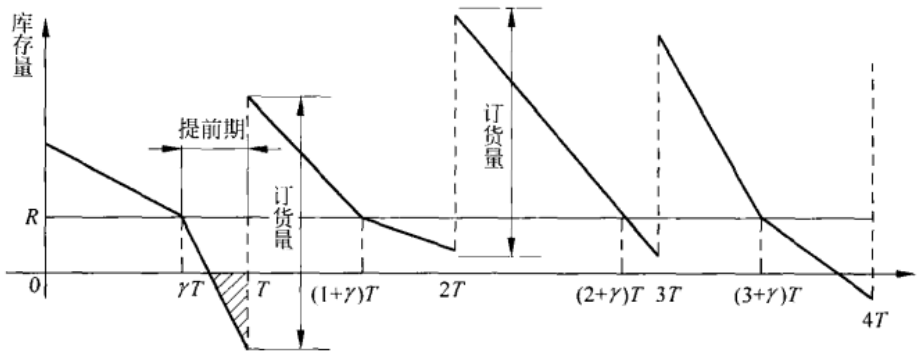


图 7.8 随机库存系统

以上讨论的随机库存系统仍然是非常简单的。对许多实际的随机系统而言，难以得到其解析的表达式，因而也就很难对其性能进行定量评价，而往往只能通过仿真的手段。

7.4.4 随机库存系统仿真举例

某公司经销单类产品，顾客需求时间均值为 0.1 个月的指数随机变量，需求量也是随机变量，其概率质量函数为

$$D = \begin{cases} 1, & \text{概率为}\frac{1}{6} \\ 2, & \text{概率为}\frac{1}{3} \\ 3, & \text{概率为}\frac{1}{3} \\ 4, & \text{概率为}\frac{1}{6} \end{cases} \tag{7.31}$$

该公司的订货策略是，按月订货，每月月初检查库存水平，若库存水平 I 超过下限 l，则不订货，若低于下限则订货，订货量为库存上限 S 与 I 之间，记订货量为 Z，则

$$Z = \begin{cases} S - I, & I < l \\ 0, & I \geqslant l \end{cases} \tag{7.32}$$

若订货，则从订货到货物入库的时间是 0.5 至 1.0 个月之间的均匀分布的随机变量。

现需要比较表 7.4 中的 9 种订货策略，以便确定何种策略费用最少。

表 7.4 库存系统的 9 种订货策略

l	20	20	20	20	40	40	40	60	60
S	40	60	80	100	60	80	100	80	100

模型中考虑如下几种费用：

(1) 订货费：设每件订货费用为 m，订货附加费用为 K（若无订货，则 $K=0$），则每月订货费为

$$C_1 = K + mZ \tag{7.33}$$

(2) 保管费：用 h 表示每件每月的保管费，显然，只有当库存水平 $I(t)>0$ 时才需要计算保管费

$$C_2 = \int_0^n hI(t)\mathrm{d}t/n \tag{7.34}$$

其中，n 为仿真运行的月数，C_2 为平均每月的保管费。

(3) 缺货损失费：用 p 表示每件缺货损失费，显然，只有当 $I(t)<0$ 时才需计算，则平均每月缺货损失费为

$$C_3 = \int_0^n p \mid I(t) \mid \mathrm{d}t/n \tag{7.35}$$

下面讨论如何对该系统建立仿真模型。

1. 随机变量模型

如前所述，需求时间随机变量服从均值为 0.1 个月的指数分布，即其概率密度函数为

$$f(D_m) = \frac{1}{0.1}\mathrm{e}^{-D_m/0.1} = 10\mathrm{e}^{-10D_m} \quad (D_m \geqslant 0) \tag{7.36}$$

其中，D_m 为需求时间间隔，可用上一节讨论过的反变换法得到，即

$$D_m = -0.1\ln u \tag{7.37}$$

随机变量需求量 D 是离散随机变量，下面讨论如何产生 D。

先根据式(7.31)做出 D 的分布函数图如图 7.9 所示。

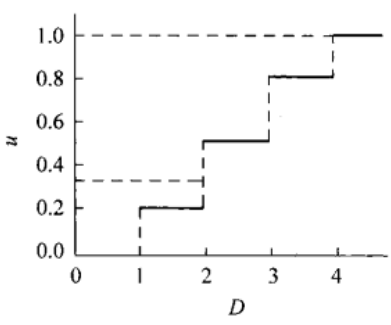


图 7.9 需求量 D 的分布函数

如果产生一个[0,1]区间上均匀分布的随机数 u，看 u 落在何区间，若 $u \leqslant \frac{1}{6}$，则令 $D=1$；如果 $\frac{1}{6} < u \leqslant \left(\frac{1}{6}+\frac{1}{3}\right)$，则令 $D=2$；而当 $\frac{1}{2} < u < \left(\frac{1}{6}+\frac{1}{3}+\frac{1}{3}\right)$ 时令 $D=3$；若 $u > \frac{5}{6}$，则令 $D=4$。显然，由于 u 是在[0,1]上均匀分布的随机变量，用它对 D 的分布函数进行取样，那么 u 落在 $\left(0,\frac{1}{6}\right]$，$\left(\frac{1}{6},\frac{1}{2}\right]$，$\left(\frac{1}{2},\frac{5}{6}\right]$，$\left(\frac{5}{6},1\right]$ 各子区间上的概率正比于上述各区间的长度，即分别为

$\frac{1}{6},\frac{1}{3},\frac{1}{3},\frac{1}{6}$，这就是我们所要求的随机变量 D。这种产生随机变量的方法称为离散反变换法。

从订货到货物入库所需时间称为订货延迟时间，它也是随机变量，用 M 来表示，服从[0.5,1]之间均匀分布，其概率密度函数为

$$f(M)=\begin{cases}\dfrac{1}{1-0.5}, & 0.5\leqslant M\leqslant 1.0\\ 0, & \text{其他}\end{cases}$$

则其分布函数为

$$F(M)=\begin{cases}0, & M<0.5\\ \dfrac{M-0.5}{1-0.5}, & 0.5\leqslant M\leqslant 1.0\\ 1, & M>1.0\end{cases}$$

参照指数分布随机变量产生的原理，也可用反变换法得到 M 为

$$M=0.5+(1-0.5)u=0.5+0.5u \tag{7.38}$$

其中，u 为[0,1]区间上均匀分布的随机数。

2. 模型中的事件定义

事件定义依赖于系统状态的描述。本系统中，库存量可描述系统的状态，那么，所订货物到达仓库，引起库存量增加，因此应定义“订货到达”为一类事件，记其类型为1；顾客需求会引起库存量减少，也应定义“需求到达”为一类事件，其类型为2。

定义了上述两类事件后，能否认为事件定义已经充分了呢？这是在事件调度法中必须十分注意的问题，从建模的观点来看，这称为模型的完备性。在事件调度法仿真建模时，必须仔细分析引起系统状态变化的各种条件及其相互关系，以保证系统状态变化描述是完备的，即能充分地对系统中的事件加以定义。分析本系统，除了货物入库及需求到达会引起库存量发生变化外，其他似乎不需要考虑了；实际上，发生“货物入库”的条件是必须首先订货，因此，还应定义“订货”为一类事件，其类型记为4。那么订货事件有什么特点呢？根据模型要求，在每月月初时要进行库存计算，如果库存量大于下限，则不需要订货，只有库存量小于下限时才需订货，因此订货事件是条件事件，它既与时间有关（每月月初），又与库存量有关，那么订货事件的完整描述应为，每月月初进行库存计算，根据库存量及订货策略决定是否订货。

在7.1节中曾经提到过，在仿真模型中，可将仿真控制定义为程序事件。在本模型中，可以将仿真运行长度定义为程序事件，设对每种策略仿真运行120个月，则当仿真钟的值等于120个月时该事件发生，定义为第3类事件。读者可能要问，为什么要将程序事件定义为第3类事件呢？这是事件调度法中另一个十分重要的问题，即所谓的“解结规则”问题。一个复杂的系统中，总是存在许多类事件，由于事件发生时间的随机性，那么就有可能出现多个事件同时发生的情形，用户在建模时必须规定同时发生事件的处理顺序，一般可按事件类型数从低到高的顺序来处理（这样便于程序的处理）。本系统中，“仿真运行120个月结束”

的程序事件与“每月月初计算库存以决定订货”的系统事件可能同时发生，那么应先执行“仿真结束”事件，因为此时再计算库存已无实际意义。

3. 确定统计变量

该系统的仿真目的是要比较 9 种不同的库存策略，评价指标是每月平均总费用 C，它包括每月平均订货费用 C_1，每月平均保管费用 C_2 及每月平均缺货损失费用 C_3。

通过上述对模型的定义，我们不难编制出仿真程序，然后进行仿真运行。各种库存策略的仿真结果见表 7.5。

```
=======Single Product Inventory System======
        Initial Inventory Level:        60 Items
        Number of Demand Sizes:         4 Items
     *** Distribution Function of Demand Sizes***
          0.167 0     0.500 0     0.833 3     1.000
        Mean Inter-Demand Time 0.1 Months
    Length of the Simulation            120 Months
     K=32.00$   I=3.00$   H=1.00$   PI=5.00$
```

表 7.5 各种库存策略的仿真结果

策略	总费用	平均订货费	平均保管费	平均缺货损失费
(20,40)	125.54	96.45	8.84	20.25
(20,60)	116.84	86.02	17.82	13.01
(20,80)	121.45	84.67	26.30	10.49
(20,100)	124.65	79.89	39.65	5.11
(40,60)	127.97	101.69	24.92	1.36
(40,80)	124.75	88.19	35.40	1.16
(40,100)	131.50	84.59	45.95	0.95
(60,80)	140.90	94.37	46.60	0.02
(60,100)	148.29	92.96	55.33	0.00

根据上述仿真结果，采用第 2 种库存策略，总平均费用最小(116.84$)，但是，需要说明的是，上述仿真结果是在以下仿真条件下得到的：各种策略采用单一的随机数据流(即共用一个随机数发生器)，仿真长度为 120 个月。如果这些条件发生变化，比如说，每种库存策略采用不同的独立的随机数发生器，或者仿真运行长度不是 120 个月，仿真结果可能会发生某些变化，甚至可能有很大的不同。关于随机数对仿真结果的影响、仿真长度如何选择、如何比较不同策略的优劣，这些内容将在第 10 章中进行讨论。

7.5 离散事件系统仿真研究的一般步骤

离散事件系统仿真研究的一般步骤与连续系统仿真是类似的，它包括系统建模，确定仿真算法，建立仿真模型，设计仿真程序，运行仿真程序，然后输出仿真结果并进行分析，一般

可用图7.10所示的流程来描述。下面就离散事件系统仿真中的一些特殊问题进行讨论。

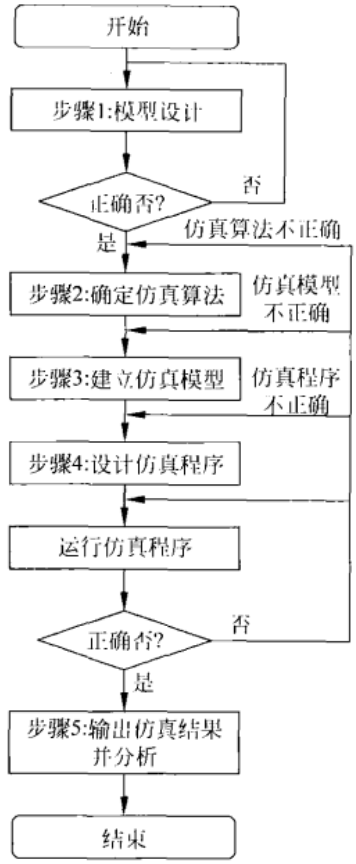


图7.10 离散事件系统仿真的一般步骤

1. 模型设计

离散事件系统的模型一般可以用流程图或网络图的方式来描述。它们都反映了临时实体在系统内部历经的过程,永久实体对临时实体的作用以及它们相互之间的逻辑关系。第8章将讨论如何由观测数据确定随机变量的分布和参数。

2. 确定仿真算法

离散事件系统仿真算法包括两方面的内容,其一是如何产生所需求的随机变量,这部分内容将在第8章中讨论;其二是采用怎样的方法对离散事件系统进行仿真,称为仿真策略,将在第9章中进行讨论。

3. 建立仿真模型

根据已确定的仿真算法,建立被仿真系统的计算机模型(变量定义及程序流程)。它是系统状态转移的动态描述,因此首先要定义系统的状态变量,这要根据系统的内部结构及仿真研究的目的来确定。即使是同一系统,仿真研究的目的不同,系统状态定义也可能不尽相同。

在离散事件系统中,状态的变化是由事件引起的,因此,要在定义系统状态的基础上定义系统事件及其有关属性。以事件调度法为例,事件类型及发生时间是其必要的属性之一,其他还应包括对事件处理的规则,是FIFO,还是LIFO,或是别的什么规则,以便按实际系统要求进行处理。在活动扫描法及进程交互法中,还要定义活动及进程,以便按活动或进程的观点来建立仿真模型。

仿真钟是仿真模型中必不可少的部件,它的推进方法决定于仿真算法,关于仿真钟在各种算法中的推进方法将在第9章中讨论。

4. 设计仿真程序

仿真程序是仿真模型的实现。如果采用仿真语言编程,则要掌握这些语言提供的各种语句。第9章中将简单介绍几类常见的仿真语言,并简要地进行评述。

5. 仿真结果分析

由于离散事件系统固有的随机性,每次仿真运行所得到的结果仅仅是随机变量的一次取样,那么,仿真结果的可信性如何?如何提高仿真结果的置信度?这些问题在这类系统仿真中占有突出的地位。有关仿真结果分析的内容将在第10章中讨论。

7.6 小结

离散事件系统是状态只在离散时间点上发生变化的系统，而且这些离散时间点一般是不确定的。由于离散事件系统这种固有的随机性，对这类系统的研究往往十分困难。经典的概率及数理统计理论、随机过程理论虽然为研究这类系统提供了理论基础，并能对一些简单系统提供解析解，但对工程实际中的大量系统，唯有依靠计算机仿真技术才能提供较为完整的结果。

本章简要介绍了实体、事件、活动、进程等仿真基本概念，重点介绍了作为离散事件系统仿真基础的仿真钟概念以及仿真钟推进的两类方法，即事件调度法和固定增量推进方法。

离散事件系统还有很多，如订票系统、库存系统、加工制造系统、交通系统、计算机系统、网络系统等。本章为了使读者对离散事件系统仿真有一个初步的理解，讨论了两类最常见的离散事件系统——排队系统及库存系统的仿真。

离散事件系统仿真研究的一般步骤与连续系统仿真是类似的，本章介绍了离散事件系统仿真的基本步骤，同时，给出了后面各章离散事件系统仿真内容在仿真步骤中的地位和作用，使读者对该类仿真有了一个总体认识。

习题

1. 试说明离散事件系统仿真与连续系统仿真的区别。

2. 对下列系统的若干实体、属性、活动、事件和状态变量命名：

①小器具维修店；②咖啡店；③杂货店；④自助洗衣店；⑤快餐店；⑥医院急诊室；⑦拥有 10 辆出租车的出租车公司；⑧汽车装配线。

3. 请给出交通干道十字路口的仿真步骤，目标是改进当前的交通流状况。考虑图 7.10 中仿真流程中，步骤 1 和步骤 2 反复 3 次，增加顺序的复杂性的问题。

4. 阅读一篇关于仿真应用方面的最新论文，该论文属于你研究的领域或你感兴趣的领域，写一份关于如何完成图 7.10 中仿真步骤的报告。

5. 有如下排队系统，试画出系统中顾客排队的队长随时间变化的情况，并统计计算仿真运行长度为 40min 时系统中顾客排队的平均队长和平均等待时间。顾客到达的时间间隔分别为 $A_i=5,6,7,14,6$(单位：min，i 表示到达顾客的顺序号)，为第 i 个顾客服务的时间分别为 $S_i=12,5,13,4,9$(单位：min)。

6. 一库存系统，一年的总订货量为 3 000 件，初始值为 100 件，每月的消耗量相等(按 25 天计算)，消耗速度相同，按月订货，每月缺货的天数允许为 3 天，提前期为 5 天，试画出库存随时间变化的曲线；若每件货物的保管费为 1 元，每次订货费为 5 元，每件货物短缺引起的损失费为 2 元，试解析计算出全年的总费用及订货点库存水平。

7. 银行只有一个取款机和只能容纳一个等待顾客的房间。当队列没有人的时候有顾

客到达将会直接进入银行办理业务。到达间隔时间和服务时间分布如下：

到达间隔时间/min	概率	服务时间/min	概率
0	0.09	1	0.2
1	0.17	2	0.4
2	0.27	3	0.28
3	0.20	4	0.12
4	0.15		
5	0.12		

仿真10个新顾客的取款机的运行情况。前10个新顾客在随机确定的时间到达。仿真开始的时候有一个顾客正在接受服务，将在时刻3离开，并且有一个顾客在队列里面等。有多少顾客进入银行交易？

参考文献

[1] (美)Law A M等著. 惠益民等译. 模拟系统的建模与分析. 北京：清华大学出版社，1987

[2] Hooper J W. Strategy-related Characteristics of discrete-event language and models. SIMULATION, 1986

[3] Pritsker A A B and Pedden C D. Introduction to Simulation and SLAM Ⅱ, Systems Publishing. West Lafayette, Ind, 1979

[4] Banks J, Carson J S著. 肖田元，范文慧译. 离散事件系统仿真. 北京：机械工业出版社，2007

[5] 王维平等. 离散事件系统建模与仿真. 北京：科学出版社，2007

[6] Kelton W D and Sadowski R P著. 周泓译. 使用Arena软件仿真. 北京：机械工业出版社，2007

第 8 章
随机变量模型的确定与产生

8.1 引言

离散事件系统一般会有一个以上的随机变量。如在单服务台系统中，顾客的到达时间间隔是随机变量，服务员为顾客服务的时间也是随机变量。

对具有随机变量的系统进行仿真，首先必须确定其随机变量的概率分布，以便在仿真模型中对这些分布进行取样得到所需要的随机变量。确定随机变量模型的基础是要搜集该随机变量的观测数据。本章仅讨论如何根据观测数据来确定概率分布的类型、参数以及如何对这种拟合进行优良度检验。

关于如何对随机变量搜集观测数据的内容，请读者参阅有关文献或书籍。

产生随机变量的基础是产生[0,1]区间上均匀分布的随机变量，亦称为随机数发生器。其他各类分布，如正态分布、γ 分布、β 分布、泊松分布等，都可以通过用某种方法对这种均匀分布进行某种变换来实现。

8.2 节讨论确定随机变量模型的有关内容；8.3 节介绍两种拟合优良度检验方法；8.4 节中将介绍两类随机数发生器，即线性同余发生器及组合发生器；并在 8.5 节中讨论随机数发生器均匀性及独立性检验的方法，这是因为这两种特性将极大地影响仿真结果的有效性；8.6 节讨论产生一般随机变量的原理；然后在 8.7 节中分别介绍典型连续随机变量及离散随机变量产生的方法。

8.2 随机变量模型的确定

根据对随机变量特性的了解程度，在确定其模型时，一般会碰到三种情形：①随机变量分布的类型已知，需要由观测数据确定该分布的参数；②由观测数据确定随机变量概率分

布类型，并在此基础上确定其参数；③由已有的观测数据难以确定该随机变量的理论分布形式，则定义一个实验分布。下面分别进行讨论。

8.2.1 分布参数的确定

1. 分布参数的类型

定义分布所采用的大多数参数，根据其物理或几何解释，可分为三个基本类型。它们是：

(1) 位置参数(记为 γ)

位置参数 γ 确定了一个分布函数取值范围的横坐标。当 γ 改变时，相应的分布函数仅仅向左或向右移动而不发生其他变化，因而又称为位移参数。

例如，均匀分布函数 $U(a,b)$，其密度函数为

$$f(x)=\begin{cases}\dfrac{1}{b-a}, & a\leqslant x\leqslant b\\ 0, & \text{其他}\end{cases} \tag{8.1}$$

其中，参数 a 定义为位置参数，当 a 改变时(保持 $b-a$ 不变)，$f(x)$ 向左或向右移动(参见图 8.1)。

(2) 比例参数(记为 β)

比例参数决定分布函数在其取值范围内取值的比例尺。β 的改变只压缩或扩张分布函数，而不会改变其基本形状。

例如，指数分布函数 $\mathrm{Exp}(\beta)$，其密度函数为

$$f(x)=\begin{cases}\dfrac{1}{\beta}\mathrm{e}^{-x/\beta}, & x\geqslant 0\\ 0, & \text{其他}\end{cases} \tag{8.2}$$

图 8.2 给出了 $\beta=2,1,0.5$ 时 $f(x)$ 的图形。

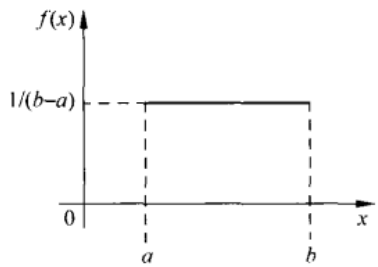


图 8.1 均匀分布 $U(a,b)$ 的密度函数

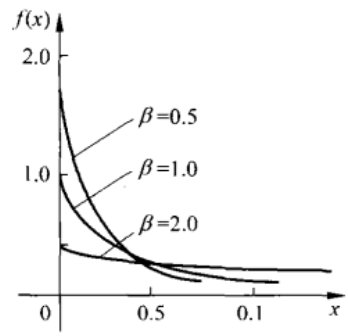


图 8.2 指数分布 $\mathrm{Exp}(\beta)$ 的密度函数

(3) 形状参数(记为 α)

形状参数确定分布函数的形状，从而改变分布函数的性质，例如，韦伯分布 $\mathrm{Weibull}(\alpha,\beta)$，其密度函数为

$$f(x)=\begin{cases}\alpha\beta^{-\alpha}x^{\alpha-1}e^{-(x/\beta)^{\alpha}}, & x>0\\ 0, & \text{其他}\end{cases} \tag{8.3}$$

当 α 改变时，其形状发生很大的变化。图 8.3 中给出了 $\alpha=1,2,3$ 时 $f(x)$ 的图形。

随机变量 X,Y，如果存在一个实数 γ，使 X 与 Y 具有相同的分布，则称 X 与 Y 仅仅是位置上不同的变量；如果对于某个正实数 β，使得 βX 与 Y 具有相同的分布，则称 X 与 Y 仅仅是比例尺不同的随机变量；如果能找到实数 γ 与 β，使 $\gamma+\beta X$ 与 Y 具有相同的分布，则称 X 与 Y 仅在位置与比例上不同。如果不存在 γ 与 β 使 $\gamma+\beta X$ 与 Y 的分布相同，则 X 与 Y 或者其形状参数不同，或者根本不服从同一类分布。

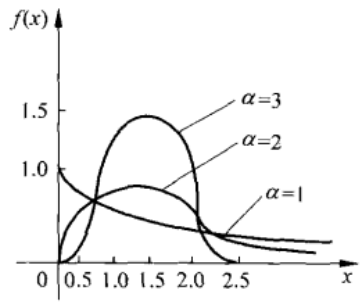


图 8.3 韦伯分布 Weibull(α,β) 的密度函数

2. 分布参数的估计

由观测数据估计某一分布参数的方法很多，使用最普遍的是最大似然估计，其他还有最小二乘估计、无偏估计等。本节我们讨论最大似然估计，其他方法可参阅有关文献。

先讨论只有一个未知参数的情形，设该参数为 θ，观测数据为 $x_1,x_2,\cdots,x_n$。

在离散分布情形，可令 $P_\theta(x)$ 为该分布的概率质量函数，定义似然函数 $L(\theta)$ 为

$$L(\theta)=P_\theta(x_1)P_\theta(x_2)\cdots P_\theta(x_n) \tag{8.4}$$

则 $L(\theta)$ 是联合质量函数，θ 的最大似然估计值 $\hat{\theta}$ 是使 $L(\theta)$ 取最大值的 θ，即对于所有可能的 θ 值，$L(\hat{\theta})\geqslant L(\theta)$。

在连续分布情形，令 $f_\theta(x)$ 为该分布的概率密度函数，其似然函数定义为 $L(\theta)$

$$L(\theta)=f_\theta(x_1)f_\theta(x_2)\cdots f_\theta(x_n) \tag{8.5}$$

下面通过两个例子来说明如何由观测值估计分布参数。

例 8.1 指数分布，被估计的参数 $\theta=\beta(\beta>0)$，其分布密度函数为 $f_\beta(x)=\frac{1}{\beta}e^{-x/\beta}$。

由

$$L(\beta)=\left(\frac{1}{\beta}e^{-x_1/\beta}\right)\left(\frac{1}{\beta}e^{-x_2/\beta}\right)\cdots\left(\frac{1}{\beta}e^{-x_n/\beta}\right)=\beta^{-n}\exp\left(-\frac{1}{\beta}\sum_{i=1}^{n}x_i\right)$$

为求使 $L(\beta)$ 取最大值的 $\hat{\beta}$，先对 $L(\beta)$ 取自然对数

$$R(\beta)=\ln L(\beta)=-n\ln\beta-\frac{1}{\beta}\sum_{i=1}^{n}x_i$$

由于 $R(\beta)=\ln L(\beta)$ 是严格递增的，$L(\beta)$ 取最大值等价于 $R(\beta)$ 取最大值，为此，对 $R(\beta)$ 求极值

$$\frac{dR}{d\beta}=-\frac{n}{\beta}+\frac{1}{\beta^2}\sum_{i=1}^{n}x_i=0$$

可得

$$\beta=\sum_{i=1}^{n}x_i/n=\bar{x}(n)$$

又由

$$\frac{d^2R}{d\beta^2}=\frac{n}{\beta^2}-\frac{2}{\beta^3}\sum_{i=1}^{n}x_i$$

当 $\beta=\bar{x}(n)$ 时，由于 x_i 为正，可见 $\dfrac{d^2R}{d\beta^2}<0$，因而 $\beta=\bar{x}(n)$ 为最大值，从而得到参数 β 的最大似然估计值为

$$\hat{\beta}=\bar{x}(n)=\sum_{i=1}^{n}x_i/n$$

■

例 8.2 泊松分布，被估计参数 $\theta=\lambda(\lambda>0)$，分布质量函数为 $P_\lambda(x)=e^{-\lambda}\lambda^x/x!$ $(x=0,1,2,\cdots)$。

由

$$L(\lambda)=P_\lambda=P_\lambda(x_1)P_\lambda(x_2)\cdots P_\lambda(x_n)=e^{n\lambda}\lambda^{\sum_{i=1}^{n}x_i}\Big/\prod_{i=1}^{n}(x_i!)$$

参照例 8.1 中相同的做法，令

$$R(\lambda)=\ln L(\lambda)=-n\lambda+\Big(\sum_{i=1}^{n}x_i\Big)\ln\lambda-\ln\Big(\prod_{i=1}^{n}(x_i!)\Big)$$

则

$$\frac{dR}{d\lambda}=-n+\frac{1}{\lambda}\sum_{i=1}^{n}x_i=0,\quad \lambda=\sum_{i=1}^{n}x_i/n=\bar{x}(n)$$

又由

$$\frac{d^2R}{d\lambda^2}=-\frac{1}{\lambda^2}\sum_{i=1}^{n}x_i<0$$

所以泊松分布参数 λ 的最大似然估计值为

$$\hat{\lambda}=\bar{x}(n)=\sum_{i=1}^{n}x_i/n$$

■

对于多个参数估计问题，上述方法原则上仍然是适用的，只是计算上要烦琐些，我们就不再举例加以说明了。

在有些情况下，分布参数的似然函数难以解析得到，此时则要采用数值计算方法。表 8.1 及表 8.2 分别列出了常见连续分布及离散分布参数的最大似然估计的计算公式。

表 8.1 连续分布参数最大似然估计

名 称	密度函数	参 数	最大似然估计
均匀分布 U(a,b)	$f(x)=\begin{cases}\dfrac{1}{b-a}, & a\leqslant x\leqslant b\\ 0, & \text{其他}\end{cases}$	位置参数 a 比例参数 $b-a$	$\hat{a}=\min\limits_{1\leqslant i\leqslant n}x_i$ $\hat{b}=\max\limits_{1\leqslant i\leqslant n}x_i$
指数分布 Exp(β)	$f(x)=\begin{cases}\dfrac{1}{\beta}e^{-x/\beta}, & x\geqslant 0\\ 0, & \text{其他}\end{cases}$	比例参数 β	$\hat{\beta}=\bar{x}(n)$
γ 分布 Gama(α,β)	$f(x)=\begin{cases}\dfrac{\beta^{-\alpha}x^{\alpha-1}e^{-x/\beta}}{\Gamma(\alpha)}, & x>0\\ 0, & \text{其他}\end{cases}$ $\Gamma(\alpha)=\int_0^{+\infty}t^{\alpha-1}e^{-t}dt$	形状参数 α 比例参数 β	$\ln\hat{\beta}+\bar{\psi}(\hat{\alpha})=\sum\limits_{i=1}^{n}\ln x_i/n!$ $\hat{\alpha}\hat{\beta}=\bar{x}(n)$ $\bar{\psi}(\hat{\alpha})=\Gamma'(\hat{\alpha})/\Gamma(\hat{\alpha})$ $\Gamma'(\hat{\alpha})$ 是 $\Gamma(\hat{\alpha})$ 的导数

续表

名　称	密度函数	参　数	最大似然估计
韦伯分布 Weibull(α,β)	$f(x)=\begin{cases}\alpha\beta^{-\alpha}x^{\alpha-1}e^{-(x/\beta)^{\alpha}}, & x>0\\ 0, & \text{其他}\end{cases}$	形状参数 α 比例参数 β	$\dfrac{\sum_{i=1}^{n}x_i^{\hat\alpha}\ln x_i}{\sum_{i=1}^{n}x_i^{\hat\alpha}}-\dfrac{1}{\hat\alpha}=\dfrac{\sum_{i=1}^{n}\ln x_i}{n}$ $\hat\beta=\left(\dfrac{\sum_{i=1}^{n}x_i^{\hat\alpha}}{n}\right)^{1/\hat\alpha}$
正态分布 $N(\mu,\sigma^2)$	$f(x)=\dfrac{1}{\sqrt{2\pi\sigma^2}}e^{-(x-\mu)^2/2\sigma^2}$ x 为实数	位置参数 μ 比例参数 σ	$\hat\mu=\bar x(n)$ $\hat\sigma=\left(\dfrac{n-1}{n}S^2(n)\right)^{1/2}$
对数正态分布 $LN(\mu,\sigma^2)$	$f(x)=\begin{cases}\dfrac{1}{x\sqrt{2\pi\sigma^2}}\exp\left(\dfrac{-(\ln x-\mu)^2}{2\sigma^2}\right), & x>0\\ 0, & \text{其他}\end{cases}$	形状参数 σ 比例参数 μ	$\hat u=\dfrac{\sum_{i=1}^{n}\ln x_i}{n}$ $\hat\sigma=\left[\dfrac{\sum_{i=1}^{n}(\ln x_i-\hat u)^2}{n}\right]^{1/2}$
β 分布 Beta(α_1,α_2)	$f(x)=\begin{cases}\dfrac{x^{\alpha_1-1}(1-x)^{\alpha_2-1}}{B(\alpha_1,\alpha_2)}, & 0<x<1\\ 0, & \text{其他}\end{cases}$ $B(\alpha_1,\alpha_2)=\int_0^1 t^{\alpha_1-1}(1-t)^{\alpha_2-1}dt$	形状参数 α_1,α_2	$\psi(\hat\alpha_1)-\psi(\hat\alpha_1+\hat\alpha_2)=\ln G_1$ $\psi(\hat\alpha_2)-\psi(\hat\alpha_1+\hat\alpha_2)=\ln G_2$ $\psi(\alpha)=\Gamma'(\hat\alpha)/\Gamma(\hat\alpha)$ $G_1=\left(\prod_{i=1}^{n}x_i\right)^{1/n}$ $G_2=\left(\prod_{i=1}^{n}(1-x_i)\right)^{1/n}$ $G_1+G_2\leqslant 1$ $\Gamma(\hat\alpha)=\int_0^{\infty}t^{\hat\alpha-1}e^{-t}dt$ $\Gamma'(\hat\alpha)$ 是 $\Gamma(\hat\alpha)$ 的导数

表 8.2　离散分布参数最大似然估计

名　称	质量函数	参　数	最大似然估计
伯努里分布 Bernoulli(p)	$P(x)=\begin{cases}1-p, & x=0\\ p, & x=1\\ 0, & \text{其他}\end{cases}$	$p\in(0,1)$	$p=\bar x(n)$
离散均匀分布 Du(i,j)	$P(x)=\begin{cases}\dfrac{1}{j-i+1}, & x\in\{i,j+1,\cdots,j\}\\ 0, & \text{其他}\end{cases}$	位置参数 i 比例参数 $j-i+1$	$\hat i=\min\limits_{1\leqslant k\leqslant n}x_k$ $\hat j=\max\limits_{1\leqslant k\leqslant n}x_k$
二项分布 Bin(t,p)	$P(x)=\begin{cases}\binom{t}{x}p^x(1-p)^{t-x}, & x\in\{0,1,\cdots\}\\ 0, & \text{其他}\end{cases}$	t 为正整数 $p\in(0,1)$	若 t 已知，$p=\bar x(n)/t$ 若 t 未知，则较困难

续表

名　称	质 量 函 数	参　数	最大似然估计
几何分布 Geom(p)	$P(x)=\begin{cases}p^x(1-p), & x\in\{0,1,\cdots\}\\ 0, & \text{其他}\end{cases}$	$p\in(0,1)$	$p=\dfrac{1}{\bar{x}(n)+1}$
负二项分布 Negbin(s,p)	$P(x)=\begin{cases}\binom{s+x-1}{x}p^x(1-p)^x, & x\in(0,1)\\ 0, & \text{其他}\end{cases}$	s 为正整数 $p\in(0,1)$	若 s 已知，$p=s/[\bar{x}(n)+s]$ 若 s 未知，则较困难
泊松分布 Poisson(λ)	$P(x)=\begin{cases}\dfrac{e^{\lambda}\lambda^x}{x!}, & x\in\{0,1,\cdots\}\\ 0, & \text{其他}\end{cases}$	$\lambda>0$	$\hat{\lambda}=\bar{x}(n)$

8.2.2　分布类型的假设

由观测数据来确定随机变量的分布类型，最常用的方法是对观测数据进行适当的预处理，然后根据预处理的结果对分布类型进行假设。

本节分别就连续分布及离散分布的观测数据的预处理方法进行讨论，并在此基础上进行分布类型的假设。

1. 连续分布类型的假设

若观测数据来自连续分布，最常用的预处理方法有三种，即点统计法、直方图法及概率图法。

(1) 点统计法

点统计法是基于连续分布的变异系数特征来进行分布类型的假设。观测数据的预处理则是计算其变异系数。变异系数的定义是

$$\delta=\sqrt{\mathrm{Var}(x)}/E(x) \tag{8.6}$$

其中，$\mathrm{Var}(x)$与 $E(x)$分别为分布的方差与均值。

例如，指数分布 $\mathrm{Exp}(\beta)$，无论 β 为何值，$\delta=1$，对于 γ 分布 $\mathrm{Gama}(\alpha,\beta)$，$\delta=1/\sqrt{\alpha}$，那么若 $\alpha<1$，则 $\delta>1$；若 $\alpha>1$，则 $\delta<1$；而 $\alpha=1$ 就是指数分布。

表 8.3 给出了常见分布的 δ 值及其取值范围。

表 8.3　连续分布的变异系数

分 布 类 型	δ	δ 取值范围
$\mathrm{U}(a,b)$	$(b-a)/[\sqrt{3}(a+b)]$	$(-\infty,\infty)$除 0 以外
$\mathrm{Exp}(\beta)$	1	1
$\mathrm{Gama}(\alpha,\beta)$	$1/\sqrt{\alpha}$	>1 $(\alpha<1)$ $=1$ $(\alpha=1)$ <1 $(\alpha>1)$

续表

分布类型	δ	δ 取值范围
Weibull(α,β)	$\left\{\dfrac{\Gamma\left(\frac{2}{\alpha}+1\right)}{\left[\Gamma\left(\frac{1}{\alpha}+1\right)\right]^2}-1\right\}^{1/2}$	>1 ($\alpha<1$) $=1$ ($\alpha=1$) <1 ($\alpha>1$)
N(μ,σ^2)	σ/μ	$(-\infty,\infty)$ 除 0 以外
LN(μ,σ^2)	$(\sigma^2-1)^{1/2}$	$(0,\infty)$
Beta(α_1,α_2)	$\left[\dfrac{\alpha_1}{\alpha_2}(\alpha_1+\alpha_2+1)\right]^{-1/2}$	$(0,\infty)$
Triang(a,b,c)	$\dfrac{(a^2+b^2+c^2-ab-ac-bc)^{1/2}}{\sqrt{2}(a+b+c)}$	$(-\infty,\infty)$除 0 以外

点统计法对观测数据进行如下预处理

$$\bar{x}(n)=\sum_{i=1}^{n}x_i/n \tag{8.7}$$

$$S^2(n)=\sum_{i=1}^{n}[x_i-\bar{x}(n)]^2/(n-1) \tag{8.8}$$

其中，$\bar{x}(n)$与 $S^2(n)$为观测数据 $x_1,x_2,\cdots,x_n$ 的均值与方差，则 δ 的似然估计为

$$\hat{\delta}=\sqrt{S^2(n)}/\bar{x}(n) \tag{8.9}$$

然后根据 $\hat{\delta}$ 值并参照各类分布的变异数据 δ 来假设观测数据的分布类型。

点统计法虽然很简单，但从表 8.3 中可以看到，这种方法不能唯一地确定分布的类型。这是因为许多分布的变异系数的取值范围是重合的，并且 $\hat{\delta}$ 是 δ 的似然估计值，但不一定是无偏的。尽管如此，点统计法仍可作为分布假设的一种粗糙的指导性方法加以使用。

(2) 直方图法

直方图法将观测数据 $x_1,x_2,\cdots,x_n$ 的取值范围分成 k 个断开的相邻区间$[b_0,b_1),[b_1,b_2),\cdots,[b_{k-1},b_k)$，每个区间宽度相等，记为 $\Delta b=b_j-b_{j-1}(j=1,2,\cdots,k)$。对任意 j，令 g_j 表示第 j 个区间$[b_{j-1},b_j)$上观测数据的个数占整个观测数据的比例数，即设 n_j 为第 j 个区间上观测点的个数，则

$$g_j=n_j/n \quad (j=1,2,\cdots,k) \tag{8.10}$$

定义函数

$$h(x)=\begin{cases}0, & x<b_0\\ g_j, & b_{j-1}\leqslant x<b_j\\ 0, & x_k\leqslant x\end{cases} \tag{8.11}$$

做出 $h(x)$的直方图，再将该图与基本理论分布的密度函数图形进行比较(先忽略位置及比例尺的差别)，观察何种分布与 $h(x)$的图形类似，则可假设观测数据 $x_1,x_2,\cdots,x_n$ 服从该类型分布；然后再采用前面介绍的方法确定其参数。

在实际使用时，可能需要对观测数据进行适当的调整，即增加一些其值特别大或特别小的观测数据，以便与理论分布进行比较。

使用直方图法的困难在于如何确定区间长度 Δb。Δb 太大，将丢失信息，Δb 太小，则观测数据中的噪声滤除得不够(一般观测数据中总是存在一定的噪声)。因此，往往要选择不同大小的 Δb 分别做出相应的直方图，选择一个看上去是“最好的”或“最光滑”的直方图与理论分布进行比较，以便得到较为可靠的分布假设。

(3) 概率图法

直方图法是将观测数据的直方图与理论分布的密度函数进行比较，而概率图法则是将观测数据定义成一个实验分布函数，然后将它与理论分布函数进行比较再进行假设。

设观测数据 $x_1, x_2, \cdots, x_n$ 共有 m 个取值($m \leqslant n$，因为可能存在取值相同的观测点)，分别记为 $x(1), x(2), \cdots, x(m)$，实验分布函定义为

$$\widetilde{F}[x(i)] = n_i/n (i = 1, 2, \cdots, m) \tag{8.12}$$

其中，n_i 表示小于或等于 $x(i)$ 的观测数据的个数，且 $n_m = n$。为了避免由有限个观测数据得到的实验分布函数值等于 1，对上式可略加修正，可采用下式来定义

$$\widetilde{F}_n[x(i)] = (n_i - 0.5)/n \tag{8.13}$$

由于分布函数的图形特征一般没有密度函数那么明显，基本上都是 S 形，靠目视来比较何种分布函数更接近 $\widetilde{F}_n[x(i)]$ 是比较困难的。

概率图法不是采用这种直接比较的方法，而是采用所谓的“分位点”比较法，下面介绍这种方法的基本思想。

分布函数的分位点定义为：设 $0 < g < 1$，则 $x_g = F^{-1}(g)$ 称为 $F(x)$ 的分位点。

如果 $F(x)$ 与 $G(y)$ 都是分布函数，分别取不同的 g 值，相应得到不同的 (x_g, y_g)，若 $F(x)$ 与 $G(y)$ 是相同的分布函数，则由 (x_g, y_g) 形成的轨迹是斜率为 45°的直线。反过来说，如果由两个分布函数 $F(x)$ 与 $G(y)$ 按相同的一组 g 值求得各自的分位点 (x_g, y_g)，在 xoy 平面上确定 (x_g, y_g) 的轨迹，若该轨迹是一条斜率为 45°的直线，则可以确认 $F(x)$ 与 $G(y)$ 的分布是相同的。

为了假设 $\widetilde{F}_n[x(i)] = (n_i - 0.5)/n = g_i$ 的分布类型，可取 $\widetilde{F}_n[x(i)]$ 的分位点为 $x(i)$，分别对应 $\widetilde{F}_n[x(i)]$ 的值为 g_i，然后从基本理论分布中选择一种，按 g_i 分别求得其分位点 y_i，然后在 xoy 平面上画出 $(x(i), y_i)$ 的轨迹，观察是否是斜率为 45°的直线，若比较接近，则可假设观测数据的分布类型与所选分布的类型相同。

有时，$(x(i), y_i)$ 的轨迹虽然呈直线形状，但斜率却不是 45°，说明这两个分布的类型是相同的，只是位置参数和(或)比例参数不同，那么可对 $x(i)$ 进行如下变换

$$y_i' = \gamma + \beta x(i) \tag{8.14}$$

得到的 $(x(i), y_i')$ 的轨迹必然是斜率为 45°的直线。这就说明，只要分位点 $(x(i), y_i)$ 的轨迹接近直线，不管其斜率如何，观测数据的分布与所选分布的类型是相同的，其位置参数及比例参数可通过前述介绍的方法来确定。

如果 $(x(i), y_i)$ 的轨迹偏离直线很远，说明所选分布不符合要求，应考虑选择其他类型的分布来进行比较。无论选择哪一种分布，可先取其位置参数为 0，比例参数为 1(如果有的话)。

概率图法只需要判断分位点轨迹偏离线性度的程度，不会对观测数据造成信息丢失，方法简单，因而得到广泛使用。

(4) 应用举例

下面通过一个例子来说明如何运用点统计法、直方图法、概率图法对观测数据进行分布类型的假设。

例 8.3 某交通干道车辆流模型，观测 90min，共 220 辆汽车通过观测点，时间间隔 x_i 如表 8.4。

表 8.4 汽车到达时间间隔观测数据 时间间隔单位：min

时间间隔	数据个数	时间间隔	数据个数	时间间隔	数据个数	时间间隔	数据个数
0.01	8	0.25	5	0.51	3	0.88	2
0.02	2	0.26	5	0.52	3	0.90	1
0.03	3	0.27	1	0.53	2	0.93	2
0.04	6	0.28	2	0.54	2	0.95	1
0.05	10	0.29	2	0.55	2	0.97	1
0.06	4	0.30	1	0.56	1	1.03	1
0.07	10	0.31	2	0.57	2	1.05	2
0.08	4	0.32	1	0.60	1	1.06	1
0.09	2	0.35	3	0.61	2	1.09	1
0.10	9	0.36	3	0.63	2	1.10	1
0.11	5	0.37	2	0.64	1	1.11	1
0.12	4	0.38	5	0.65	3	1.12	1
0.13	2	0.39	1	0.69	2	1.17	1
0.14	4	0.40	2	0.70	1	1.18	1
0.15	6	0.41	2	0.72	3	1.24	1
0.17	1	0.43	3	0.74	1	1.28	12
0.18	1	0.44	1	0.75	1	1.33	1
0.19	3	0.45	2	0.76	1	1.38	1
0.20	1	0.46	1	0.77	1	1.44	1
0.21	5	0.47	3	0.79	1	1.51	1
0.22	3	0.48	1	0.84	1	1.72	1
0.23	5	0.49	4	0.86	1	1.83	1
0.24	1	0.50	3	0.87	1	1.96	1

① 点统计法

由表 8.4 中数据易于计算得到

$$\bar{x}(219) = 0.399(\text{min})$$

$$S^2(219) = 0.144(\text{min})$$

从而可以得到

$$\hat{\delta} = \frac{\sqrt{S^2(219)}}{\bar{x}(219)} = \frac{\sqrt{0.144}}{0.399} = 0.951$$

由于 $\hat{\delta}$ 接近 1，根据表 8.3，可以假设该观测数据服从指数分布，其参数 $\hat{\beta}=\bar{x}(219)=0.399$。

② 直方图法

将表 8.4 中的观测数据的取值范围分成若干个等长区间，每个区间的长度 Δb 分别取为 0.05，0.075，0.10，且令 $b_0=0$，得到三个直方图分别如图 8.4～图 8.6 所示。

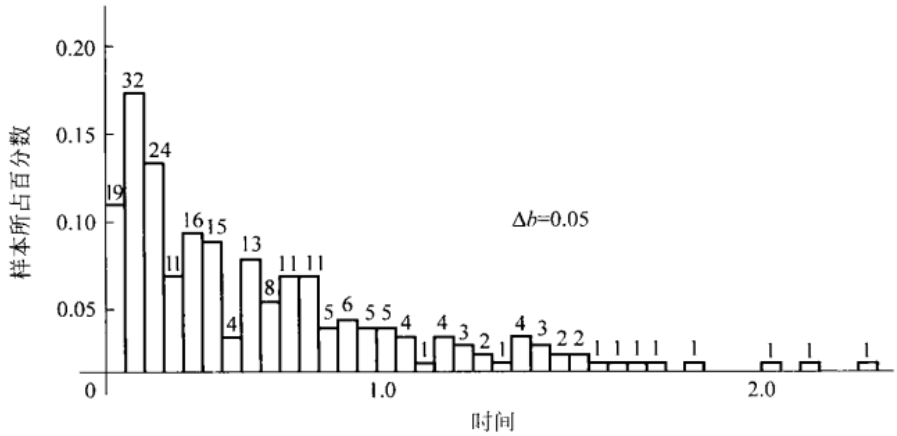


图 8.4 $\Delta b=0.05$ 时的直方图

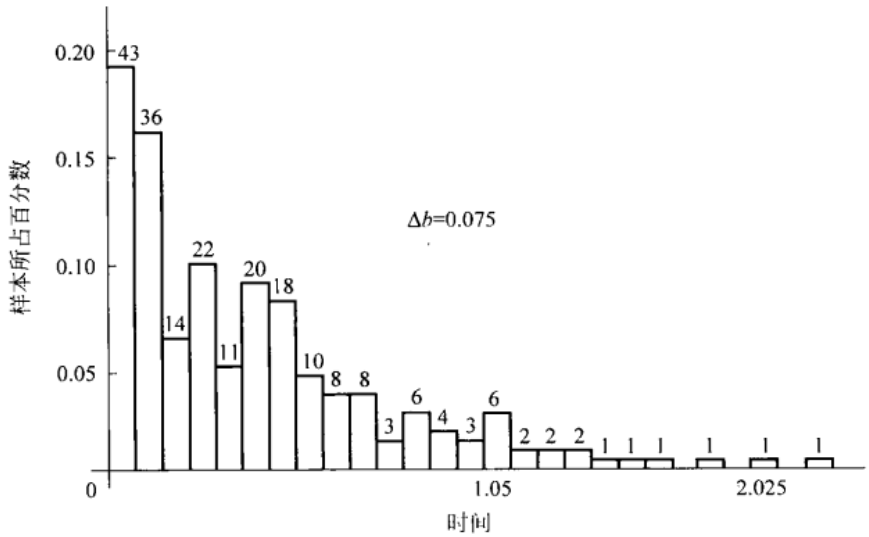


图 8.5 $\Delta b=0.075$ 时的直方图

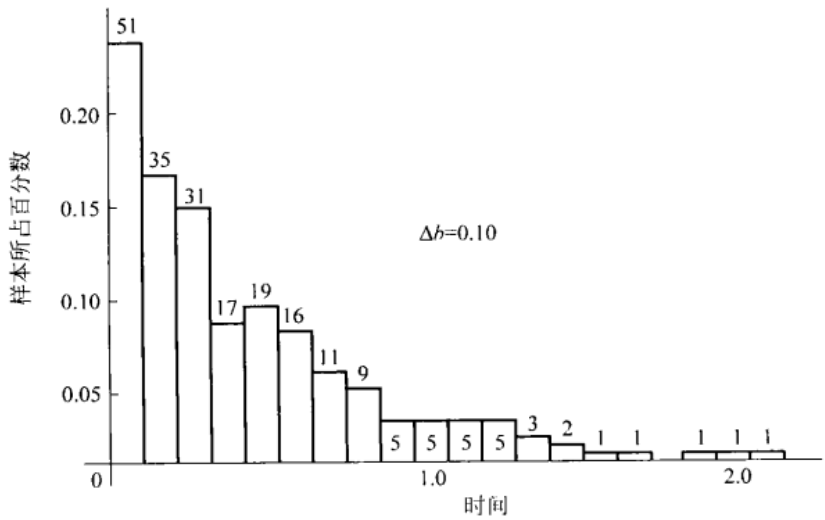


图 8.6 $\Delta b=0.10$ 时的直方图

比较上述三个图，可以看出 $\Delta b=0.10$ 时图形比较平滑，其形状确与指数分布的密度函数接近，因而可以假设该观测数据服从指数分布。

③ 概率图法

先定义观测数据的实验分布函数为

$$\widetilde{F}_{219}[x(i)]=(n_i-0.5)/219 \quad (i=1,2,\cdots,92)$$

然后，根据表 8.4 计算出 $g_i=(n_i-0.5)/219$。

再选择一个分布函数 $F(y)$，我们不妨分别选择指数分布函数 Exp(1.0)及正态分布函数 N(0,1)。按 g_i 分别对 Exp(1.0)及 N(0,1)进行逆运算，分别得到 y_{Ei} 及 y_{Ni}，其中 y_{Ei} 表示对 Exp(1.0)进行逆运算的结果，y_{Ni} 表示对 N(0,1)进行逆运算的结果($i=1,2,\cdots,92$)，然后分别做出由($x(i)$，y_{Ei})及($x(i)$，y_{Ni})的轨迹图形如图 8.7、图 8.8 所示。

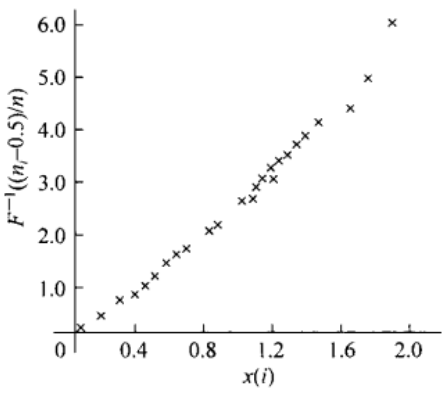


图 8.7 表 8.4 中的观测数据与 Exp(1.0)的概率图

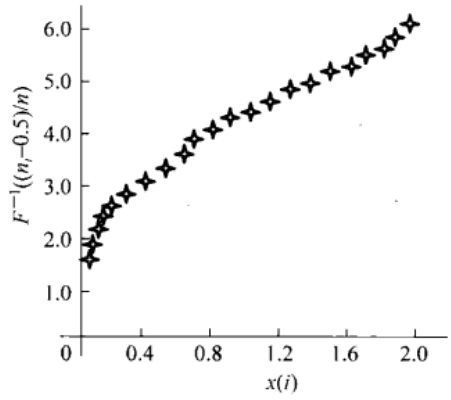


图 8.8 表 8.4 中的观测数据与 N(0,1)的概率图 ■

由图 8.7 可以看到，由($x(i)$，y_{Ei})形成的图形是相当直的，而在图 8.8 中，在其低端呈现出明显的非线性。因而，假设观测数据服从正态分布是不合理的，而假设它们服从指数分布则是比较合理的。当然，由于样本的有限性，所得图形往往不可能恰好形成一条直线。这种假设的进一步检验，则要借助于拟合优良度检验方法，将在 8.3 节中讨论。

2. 离散分布类型的假设

下面讨论用于离散分布类型假设的两种方法，即点统计法与线图法。其中线图法相应于连续分布的直方图法。由于离散分布的分布函数是不连续的，其分布函数的逆函数不易定义，因而一般不采用概率图法。

(1) 点统计法

几类常见的离散分布的变异系数 τ 的计算公式及其取值范围见表 8.5。

表 8.5 常见离散分布的变异系数

分布类型	τ	τ 的取值范围
Beroulli(p)	$1-p$	(0,1)
DU(i,j)	$[(j-i+1)^2-1]/[6(i+j)]$	$(-\infty,\infty)$
Bin(1,p)	$1-p$	(0,1)
geom(p)	$1/p$	$(1,\infty)$
negbin(s,p)	$1/p$	$(1,\infty)$
Poisson(λ)	1	1

离散情形下的点统计法与连续情形的做法是相同的，即由观测数据 $x_1, x_2, \cdots, x_n$ 计算出

$$\bar{x}(n) = \sum_{i=1}^{n} x_i / n$$

$$S^2(n) = \sum_{i=1}^{n} [x_i - \bar{x}(n)]^2 / (n-1)$$

根据 $\bar{x}(n)$ 及 $S^2(n)$ 估计出 $\hat{\tau}$ 为

$$\hat{\tau} = \sqrt{S^2(n)} / \bar{x}(n) \tag{8.15}$$

将 $\hat{\tau}$ 与理论分布的 τ 值进行比较后，就可进行分布类型的假设。例如，当 $\hat{\tau}<1$ 时，可假设为二项分布；若 $\hat{\tau}$ 接近 1，则可假设为泊松分布。

(2) 线图法

设观测数据为 $x_1, x_2, \cdots, x_n$，将其按递增顺序排列，设共有 m 个取值($m \leqslant n$)，分别为 $x(1), x(2), \cdots, x(m)$。记 $x(i)$的数据个数占整个观测数据个数的比例为 h_i，以 $x(i)$做自变量，以 h_i 的值作为函数值，即

$$h_i = f(x(i)) \quad (i = 1, 2, \cdots, m) \tag{8.16}$$

由函数值 h_i 向相应的自变量 $x(i)$做垂线所得到的图形称为线图。

线图法是将观测数据的线图与假设的理论分布的质量函数进行比较以便找到相似的图形。

由此可以看出，线图法不需要将观测数据划分新的区间，因而不会丢失任何信息，较之连续情形的直方图法使用起来更为方便。

(3) 应用举例

下面通过一个简单的例子来说明如何运用点统计法及线图法来进行离散分布的假设。

例 8.4 一库存系统，观测统计每天发送的零件数量，观测结果如表 8.6 所示，现根据该表中的数据确定零件发送模型。

表 8.6 零件发送观测数据

零件个数	天数	零件个数	天数	零件个数	天数	零件个数	天数
1	2	4	10	7	10	10	2
2	4	5	12	8	10	11	1
3	6	6	18	9	10	12	2

① 点统计法

表 8.6 中共有 87 个观测数据，不难得到其均值与方差

$$\bar{x}(87) = 6.115, S^2(87) = 25.777$$

则变异系数为

$$\hat{\tau} = \sqrt{S^2(87)} / \bar{x}(87) = 0.945$$

由于 $\hat{\tau}$ 的值很接近 1，可考虑观测数据服从泊松分布。

② 线图法

表 8.6 已将观测数据按递增顺序分组，这样易于计算 $h_i (i=1,2,\cdots,12)$，并做出其线图如图 8.9 所

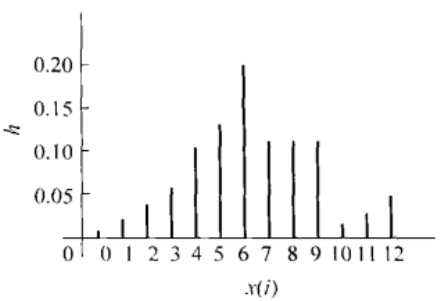


图 8.9 零件发送观测数据的线图

示。将其与泊松分布 Poisson(λ)=Poisson(6)比较，它们的形状是很接近的，可假设观测数据服从泊松分布 Poisson(6)。 ■

8.2.3 实验分布

如果由观测数据难以确定一个理论分布，则只好采用实验分布作为观测数据的模型。

原始观测数据一般有两种形式，一种是原始单个数据，另一种是分组数据。

若是原始单个数据 $x_1,x_2,\cdots,x_n$，在定义其实验分布时，先将该 n 个数据按递增顺序排列。由于可能有相同值的数据，经排序后得到 $x(1),x(2),\cdots,x(m)$，$m\leqslant n$，该观测数据的实验分布可由下式来定义

$$F(x)=\begin{cases}0, & x<x(1)\\ \dfrac{i-1}{n-1}+\dfrac{1}{n-1}\dfrac{x-x(j)}{x(j+1)-x(j)}, & x(j)\leqslant x<x(j+1)\\ & (j=1,2,\cdots,m)\\ 1, & x>x_n\end{cases}\tag{8.17}$$

若观测数据是分组数据，即不知道观测数据的数值，而仅知道该 n 个数据分布在 m 个相邻区间$[a_0,a_1),[a_1,a_2),\cdots,[a_{m-1},a_m)$上及每个区间上数据的个数。为定义这类观测数据的实验分布，记第 j 个区间上的个数为 $n_j(j=1,2,\cdots,m)$，则 $n_1+n_2+\cdots+n_m=n$，实验分布函数的表达式为

$$F(x)=\begin{cases}0, & x<a_0\\ \dfrac{\sum_{i=1}^{k-1}n_i}{n}+\dfrac{n_j}{n}\dfrac{x-a_{j-1}}{a_j-a_{j-1}}, & a_{j-1}\leqslant x<a_j\quad(j=1,2,\cdots,m)\\ 1, & x\geqslant a_m\end{cases}\tag{8.18}$$

按式(8.17)及式(8.18)定义实验分布的缺点是，随机变量取值只可能在 $x(1)\leqslant x\leqslant x(m)$或 $a_0\leqslant x\leqslant a_m$ 范围内。为了消除这种有界性，可对式(8.17)和式(8.18)略加修改，即在 $F(x)$的头尾增加一个“指数尾”，具体做法读者不妨试一试。

若观测数据是离散随机变量，在原始单个数据情形，可定义其实验分布的质量函数如下

$$P(x_i)=(x_i\text{ 的数据个数})/n\quad(i=1,2,\cdots,m)$$

其中

$$\sum_{i=1}^{m}(x_i\text{ 的数据个数})=n\tag{8.19}$$

对分组数据，其质量函数为

$$P(x)=(x\text{ 所在区间的数据个数})/n\tag{8.20}$$

然后由这些质量函数即可得到相应的实验分布函数。

例 8.5 已知观测数据 x_i 为 8，−1，2，2，0，4，6，4，7，9，11。试定义其连续实验分布函数。

将 x_i 排序，并将数据值相同的分成一组，统计每组数据的个数，按式(8.17)计算每个 $x(j)(j=1,2,\cdots,m)$对应的 $F(x(j))$的值，其结果如表 8.7 所示。

表 8.7 计算结果

i	1	2	3	4	5	6	7	8	9	10	11
x_i	−1	0	2	2	4	4	6	7	8	9	11
j	1	2	3		4		5	6	7	8	9
$x(i)$	−1	0	2		4		6	7	8	9	11
$F(x(j))$	0	0.1	0.3		0.5		0.6	0.7	0.8	0.9	1.0

从而 $F(x(j))$所定义的实验分布函数如图 8.10 所示。

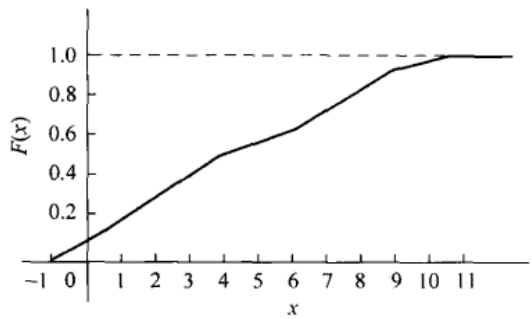


图 8.10 $F(x(j))$定义的连续实验分布函数

■

8.3 拟合优良度检验

由观测数据假设了其分布的类型并估计出其参数以后，一般需要检验该分布与这些观测数据吻合的程度，即进行拟合优良度检验，本节介绍两种常用的检验方法。

8.3.1 χ^2 检验

对拟合分布进行 χ^2 检验时，要将该拟合分布的取值范围分为 K 个相邻区间$[a_0,a_1)$，$[a_1,a_2),\cdots,[a_{k-1},a_k)$，其中可能 $a_0=-\infty$和/或 $a_k=+\infty$，然后计算

$$p_j=\int_{a_{j-1}}^{a_j}\hat{f}(x)\mathrm{d}x \quad (j=1,2,\cdots,K) \tag{8.21}$$

其中，$\hat{f}(x)$是拟合的分布密度函数。对离散情形，有

$$p_j=\sum_{(i;\,a_{j-1}\leqslant x_i<a_j)}\hat{p}(x_i) \qquad (j=1,2,\cdots,K) \tag{8.22}$$

其中，$\hat{p}(x_i)$是拟合分布的质量函数。

χ^2 检验的步骤可概括如下：

(1) 分别计算每个区间上观测数据的个数 $N_j(j=1,2,\cdots,K)$，记 $\sum_{j=1}^{K} N_j = n$。

(2) 计算按拟合分布得到的期望个数，即 $nP_j(j=1,2,\cdots,K)$。

(3) 计算 χ^2 检验的统计值

$$\chi^2 = \sum_{j=1}^{K} \frac{(N_j - nP_j)^2}{nP_j} \tag{8.23}$$

(4) 结果判断　首先要规定检验水平 α，如果拟合分布中有 m 个参数是从观测数据按最大似然估计得到的，则可以证明，当 $n\to\infty$ 时

$$\chi^2_{K-m-1,1-\alpha} \leqslant \chi^2_{1-\alpha} \leqslant \chi^2_{K-1,1-\alpha} \tag{8.24}$$

判断准则有两种：

① 若 $\chi^2_{1-\alpha} > \chi^2_{K-1,1-\alpha}$，则拒绝如下原假设 H_0：观测数据 x_i 是密度函数为 $f(x)$ 的独立同分布随机变量；否则不拒绝 H_0。

② 若 $\chi^2_{1-\alpha} \leqslant \chi^2_{K-m-1,1-\alpha}$，则不拒绝原假设 H_0，否则拒绝 H_0。

显然，第二种准则较第一种准则更严格一些。当然，当 K 较大(例如 $K>10$)，且 m 不大于 2 时，$\chi^2_{K-m-1,1-\alpha}$ 与 $\chi^2_{K-1,1-\alpha}$ 相差较小，较多采用的是第一种准则。

进行 χ^2 检验时，区间的确定将影响检验的效能。为了使检验无偏，要求按 P_j 基本相等来确定区间，即所选区间 $[a_{j-1},a_j)(j=1,2,\cdots,K)$，使 $P_1=P_2=\cdots=P_K$。另外，根据经验，区间的个数宜在 30～40 以下，并能使 $nP_j\geqslant 5$，以提高检验的有效性。当然，在离散分布的情形下，不可能保证 P_j 完全相等，但应使 P_j 的值尽可能接近。

例 8.6　在例 8.3 中，我们将观测数据拟合成 $\hat{F}(x)=1-e^{-x/0.399}(x\geqslant 0)$，现进行 $\alpha=0.10$ 水平的 χ^2 检验。

本题中，$n=219$，为保证 $nP_j\geqslant 5$，则要求 $P_j\geqslant 0.025$，取 $P_j=0.05$，$K=20$，下面确定各区间的 $a_j(j=1,2,\cdots,K)$。

由于 $\hat{F}(x)$ 的 x 的取值范围为 $[0,\infty)$，因此，$a_0=0$，$a_{20}=\infty$。对 $a_j(j=1,2,\cdots,19)$，我们可以采取以下方法

$$\int_{a_0}^{a_1} \hat{f}(x)\mathrm{d}x = (1-e^{-x/0.399})\Big|_{a_0}^{a_1} = 0.05$$

可得

$$e^{-a_1/0.399}=0.95$$

$$a_1=-0.399\ln 0.95=0.02$$

依次下去，可计算得到 $a_2,a_3,\cdots,a_{19}$。

确定区间后，则可对观测数据进行统计计算，分别得到 N_j 及 $(N_j-nP_j)^2/(nP_j)$，最后计算出 χ^2 值，上述计算的结果如表 8.8 所示。

$\chi^2=22.182$，查 $\chi^2_{19,0.90}$ 分布表可得 $\chi^2_{19,0.90}=27.204$，因此，$\chi^2_{0.90}<\chi^2_{19,0.90}$，则可以在 0.10 水平上不拒绝原假设 H_0。

表 8.8 车辆到达时间间隔模型的 χ^2 优良度检验

j	$a_{j-1}\ a_j$	N_j	$(N_j-nP_j)^2/(nP_j)$
1	[0,0.020)	8	0.795
2	[0.020,0.042)	11	0.000
3	[0.042,0.065)	14	0.850
4	[0.065,0.089)	14	0.850
5	[0.089,0.115)	16	2.329
6	[0.115,0.142)	10	0.082
7	[0.142,0.172)	7	1.425
8	[0.172,0.204)	5	3.233
9	[0.142,0.239)	13	0.384
10	[0.239,0.277)	12	0.101
11	[0.277,0.319)	7	1.425
12	[0.319,0.366)	7	1.425
13	[0.366,0.419)	12	0.101
14	[0.419,0.480)	10	0.082
15	[0.480,0.553)	20	7.480
16	[0.553,0.642)	9	0.347
17	[0.642,0.757)	11	0.000
18	[0.757,0.919)	9	0.347
19	[0.919,1.195)	14	0.850
20	[1.195,∞)	10	0.082

■

8.3.2 柯尔莫哥洛夫-斯米尔洛夫(K-S 检验)

由 8.3.1 节的讨论我们已经看到,为了提高 χ^2 检验的有效性及无偏性,要求 $nP_j\geqslant 5$,且 $P_j(j=1,2,\cdots,K)$大致相等。这就带来两方面的困难,一方面,按 P_j 相等来确定$[a_{j-1},a_j)$时要对$\hat{F}(x)$进行逆运算,而在某些情况下,求$\hat{F}(x)$的逆运算比较困难,或者$\hat{F}(x)$无封闭形式无法求$\hat{F}(x)$的逆运算;另一方面,当 n 较小时,$P_j\geqslant 5/n$ 的值较大,从而得到的区间过大,结果造成观测数据的信息丢失。

K-S 检验则可避免上述困难,它不需要对观测数据分组,而是将拟合的分布函数$\hat{F}(x)$与由观测数据定义的实验分布函数$\widetilde{F}(x)$进行比较。

在 K-S 检验中,观测数据的实验分布函数采用如下定义:

设观测数据为 $x_1,x_2,\cdots,x_n$,其实验分布函数$\widetilde{F}(x)$是

$$\widetilde{F}_n(x)=\frac{(x_i\leqslant x)\text{ 数据的个数}}{n}\qquad(\text{对所有 }x)$$

这样,$\widetilde{F}(x)$是右连续的阶跃函数。

K-S 检验是根据$\widetilde{F}(x)$与$\hat{F}(x)$的接近程度来决定是否拒绝原假设 H_0。评价接近程度的指标是采用$\widetilde{F}(x)$与$\hat{F}(x)$之间的最大距离 D_n

$$D_n = \sup_x\{|\widetilde{F}_n(x) - \hat{F}(x)|\} \tag{8.25}$$

若 D_n 超过规定的常数 $d_{n,1-\alpha}$（其中 α 是要求的检验水平），则拒绝 H_0，否则不拒绝 H_0。

遗憾的是，对于不同的分布，$d_{n,1-\alpha}$ 的值是不同的；即使是同一分布，不同的 α 下 $d_{n,1-\alpha}$ 也不相同，而且尚无通用的表可查（如同 χ^2 检验那样），这就影响了 K-S 检验使用的普遍性。下面介绍三种典型分布的 K-S 检验。

1. 指数分布 Exp($\hat{\beta}$)

由观测数据拟合的指数分布的分布函数 $\hat{F}(x)=1-e^{-x/\beta}$，其中参数 $\hat{\beta}=\bar{x}(n)$，其 K-S 检验的表达式为

$$\left(D_n - \frac{0.2}{n}\right)\left(\sqrt{n} + 0.26 + \frac{0.5}{\sqrt{n}}\right) > d_{1-\alpha} \tag{8.26}$$

若上式成立，则拒绝 H_0，其中 $d_{1-\alpha}$ 的值见表 8.9。

表 8.9 指数分布 $d_{n,1-\alpha}$ 的值

α	0.150	0.100	0.050	0.25	0.10
$d_{n,1-\alpha}$	0.926	0.990	1.094	1.190	1.308

2. 正态分布 N($\hat{\mu}$,$\hat{\sigma}^2$)

设该分布的参数 μ,σ^2 是由观测数据按最大似然估计得到的，即

$$\hat{\mu} = \bar{x}(n) \quad \hat{\sigma}^2 = S^2(n) = \sum_{i=1}^{n}(x - \bar{x}(n))/(n-1)$$

则 K-S 检验的表达式为

$$\left(n - 0.01 + \frac{0.85}{n}\right)D_n > d_{1-\alpha} \tag{8.27}$$

若上式成立，则拒绝 H_0，其中 $d_{1-\alpha}$ 的值见表 8.10。

表 8.10 正态分布 $d_{1-\alpha}$ 的值

α	0.150	0.100	0.050	0.25	0.10
$d_{1-\alpha}$	0.775	0.819	0.895	0.955	1.035

3. 韦伯分布 Weibull($\hat{\alpha}$,$\hat{\beta}$)

设韦伯分布的参数 $\hat{\alpha}$,$\hat{\beta}$ 是由观测数据得到的最大似然估计值，即由以下两式计算得到的

$$\begin{cases} \dfrac{\sum_{i=1}^{n} x_i^{\hat{\alpha}}}{\sum_{i=1}^{n} x_i^{\hat{\alpha}}} - \dfrac{1}{\hat{\alpha}} = \dfrac{\sum_{i=1}^{n}\ln x_i}{n} \\ \hat{\beta} = \left(\dfrac{\sum_{i=1}^{n} x_i^{\hat{\alpha}}}{n}\right)^{1/2} \end{cases} \tag{8.28}$$

其中 $\hat{\alpha},\hat{\beta}$ 是 α,β 的估计值，那么韦伯分布的 K-S 检验的表达式为

$$D_n > d_{n,1-\alpha} \tag{8.29}$$

若上式成立，则拒绝 H_0，其中 $d_{n,1-\alpha}$ 的值见表 8.11。

表 8.11 韦伯分布 $d_{n,1-\alpha}$ 的值

n	α				
	0.20	0.15	0.10	0.05	0.01
20	0.158	0.165	0.175	0.191	0.220
30	0.130	0.136	0.144	0.156	0.179
40	0.113	0.118	0.125	0.136	0.125

8.4 随机数发生器

严格地说，仿真中采用的随机数发生器不是在概率论意义下的真正的随机数，而只能称为伪随机数，因为无论哪一种随机数发生器都采用递推算法。尽管如此，如果算法选择得合适，由这种算法得到的数据，通过对它进行统计检验后能具有较好的统计特性（如均匀性、独立性等），则将这种伪随机数用于仿真仍然是可行的。

目前使用的大多数随机数发生器是线性同余发生器，它是 Lehmer 在 1951 年提出的；另一类是组合发生器。下面介绍这两类发生器的原理。

8.4.1 线性同余发生器

线性同余发生器采用式(8.30)产生随机数，即

$$Z_i = (aZ_{i-1} + C)(\mathrm{mod}\ m) \tag{8.30}$$

其中，Z_i 是第 i 个随机数，a 为乘子，C 为增量，m 为模数，Z_0 称为随机数源或种子，它们均为非负整数。显然，由式(8.30)得到的 Z_i 满足

$$0 \leqslant Z_i \leqslant m-1$$

为了得到[0,1]区间上所需要的随机数 U_i，可令

$$U_i = Z_i/m$$

显然，由式(8.30)得到的 Z_i 实质上完全不是随机的，因为设

$$K_i = \mathrm{mod}(aZ_{i-1}+C,m)$$

则

$$\begin{aligned} Z_1 &= aZ_0 + C - mK_1 \\ Z_2 &= a^2Z_0 + aC - maK_1 + C - mK_2 \\ &= a^2Z_0 + C(1+a) - m(K_2 + aK_1) \end{aligned}$$

$$Z_3 = a^3 Z_0 + C(1 + a + a^2) - m(K_3 + K_2 a + K_1 a^2)$$

$$\vdots$$

$$Z_n = a^n Z_0 + C(1 + a + \cdots + a^{n-1}) - m(K_n + K_{n-1} a + \cdots + K_1 a^{n-1})$$

$$= a^n Z_0 + C\frac{a^n - 1}{a - 1} (\text{mod } m)$$

即一旦 m, a, C, Z_0 确定，则 Z_i 就完全确定下来了。

另外，由于 Z_i 是$[0, m-1]$区间上的整数，那么由 Z_i 得到的 U_i 仅仅是有限个数，即为 $0, 1/m, 2/m, \cdots, m-1/m$，而不可能位于这些数值之外。

例 8.7 观察 $m=16, a=5, C=3, Z_0=7$ 的线性同余发生器，表 8.12 列出了 $i=1, 2, \cdots, 19$ 时 Z_i, U_i 的值。

表 8.12 线性同余发生器($Z_i = 5Z_{i-1} + 3 (\text{mod } 16)$ $Z_0 = 7$)

i	Z_i	U_i	i	Z_i	U_i
0	7		10	9	0.563
1	6	0.375	11	0	0.000
2	1	0.063	12	3	0.188
3	8	0.500	13	2	0.125
4	11	0.688	14	13	0.813
5	10	0.625	15	4	0.250
6	5	0.313	16	7	0.438
7	12	0.750	17	6	0.375
8	15	0.938	18	1	0.063
9	14	0.875	19	8	0.500

■

从表 8.7 中可以看到该随机数发生器的几个特点：

(1) 线性同余发生器产生的 Z_i 值确位于$[0, m-1]$区间上，因而 U_i 位于$[0, 1]$区间内。

(2) 适当选择 m, a, C，可使 Z_i 循环产生，无论 Z_0 取何值，其循环顺序是相同的。其循环一次称为发生器的一个周期，记为 P。如果 $P=m$，则称该发生器具有满周期。

(3) 适当地选择 m, a, C，可保证 Z_i 在$[0, m-1]$区间上一个周期内每个整数正好出现一次，从而保证了均匀性。

(4) 为提高 U_i 的均匀性，要求加大 m。

如果 m 足够大，且发生器具有满周期，那么可以预计，所得到的 U_i 在$[0, 1]$区间上是均匀分布的，且取值是相当密的。这样就可足够近似于真正的$[0, 1]$区间上的均匀分布 $U(0, 1)$。

那么，如何选择 m, a, C，就能保证线性同余发生器具有满周期呢？下面给出一个定理。

定理：当且仅当下列条件满足时，线性同余发生器具有满周期：

(1) m 与 C 能同时被整除的唯一正整数是 1；

(2) 如果 q 是整除 m 的素数(q 只能被自身及 1 整除)，则 q 能整除 $a-1$；

(3) 如果 m 能被 4 整除，则 $a-1$ 也能被 4 整除。

在实际使用中，线性同余发生器有两种形式，即混合乘同余($C>0$)及乘同余($C=0$)。

对混合乘同余发生器，一般选择 $m=2^b$，C 为奇数，而 $a-1$ 可被 4 整除，将得到满周期。Z_0 可为 0 到 $m-1$ 之间的任意整数而不会对随机数发生器的周期发生任何影响。

对乘同余发生器，由于 $C=0$，无论怎样选择 m，则定理的条件(1)满足不了，因而不可能得到满周期。若选择 $m=2^b$，则可以证明，随机数的周期 $P\leqslant 2^{b-2}$，即 $0\sim m-1$ 之间的整数至多只有四分之一可能成为 Z_i 的值，而且这四分之一的整数在 $0\sim m-1$ 之间是如何分布的，即是否存在一个大缺口亦难以确定，这与种子 Z_0 的值有关。若 $a=8K+3$ 或 $8K+5$，且 Z_0 为奇数，则 $P=2^{b-2}$。

为了提高乘同余的可用性，人们进行了大量研究，发现可通过恰当选择 m 及 a 实现接近满周期，较突出的是素数取模乘同余法(PMMLCG)。下面介绍这种方法。

在 PMMLCG 中，m 是小于 2^b 的最大素数，而 a 的选择满足以下条件，即 a^l-1 能被 m 整除的最小整数是 $l=m-1$，也就是说能被 m 整除的(a^l-1)的最小整数为 $a^{m-1}-1$，那么得到的 Z_i 的周期为 $m-1$，且在每个周期内，$1,2,\cdots,m-1$ 这些整数严格地只出现一次。

PMMLCG 的优点是很明显的，它避免了一般乘同余中的“缺口”问题，m 容易确定，不需要选择 C，若 m 足够大，其周期也很长。下面给出两个经过检验的、性能较好的 PMMLCG，它们是

$$Z_i = 5^5 Z_{i-1} (\mathrm{mod}\ 2^{35}-31) \tag{8.31}$$

$$Z_i = 8^5 Z_{i-1} (\mathrm{mod}\ 2^{31}-1) \tag{8.32}$$

乘同余法的出现早于混合乘同余法，因而使用得比较广泛。目前，许多仿真语言中使用的则是混合乘同余法。

8.4.2 组合发生器

为了提高线性同余发生器的性能，人们提出将两个独立的线性同余发生器组合起来，即用一个发生器控制另一个发生器产生的随机数，因而称为组合发生器。

迄今为止，有两种控制方法使用得比较广泛。第一种方法是：首先从第一个发生器产生 K 个 $Z_i(U_i)$，得到数组 $U=(U_1,U_2,\cdots,U_K)$或 $Z=(Z_1,Z_2,\cdots,Z_K)$；然后用第二个随机数发生器产生在$[1,K]$区间上均匀分布的随机整数 I；以 I 作为数组 U(或 Z)的元素下标，将 U_I 或 Z_I 作为组合发生器产生的随机数，然后从第一个发生器再产生一个随机数来取代 U_I 或 Z_I，依次下去。

第二种方法是：设 $Z_i^{(1)}$ 与 $Z_i^{(2)}$ 分别是由第一个与第二个线性同余发生器产生的随机数，则令 $Z_i^{(2)}$ 的二进制表示的数循环移位 $Z_i^{(1)}$ 次，得到一个新的位于 0 到 $m-1$ 间的整数 $Z'^{(2)}_i$；然后将 $Z_i^{(2)}$ 与 $Z'^{(2)}_i$ 的相应二进制位“异或”相加得到组合发生器的随机变量 Z_i，且令 $U_i=Z_i/m$。

组合发生器的优点是，大大减少了由式(8.30)带来的自相关，提高了独立性；还可以加长发生器的周期，提高随机数的密度，从而提高了均匀性。而且它一般对构成组合发生器的线性同余发生器的统计特性要求较低，得到的随机数的统计特性却比较好。组合发生器的缺点是速度慢，因为要得到一个随机数，需要产生两个基础的随机数，并执行一些辅助操作。

8.5 随机数发生器的测试

正如在 8.4 节中指出过的那样，计算机仿真中所采用的随机数发生器实际上是伪随机数发生器，因此，它们的性能只能在某种程度上近似[0,1]区间上均匀分布的随机数。对于给定的随机数发生器其性能如何，在使用之前进行检验是十分必要的。

8.5.1 均匀性检验

评价随机数发生器的性能指标之一是均匀性，即要求由随机数发生器产生的伪随机数 U_i 能“均匀”地分布在[0,1]区间内。

最常使用的均匀性检验方法是频率检验，其做法是：将随机数发生器的取值范围[0,1]分成 K 个互不重叠的等长的子区间，然后由该随机数发生器产生 N 个随机数 $U_i(i=1,2,\cdots,N)$。按照均匀性的要求，随机数落在每一个子区间上的概率 $P=\frac{1}{K}$，即落在每一个子区间上的随机数个数的理论值为 $n=N/K$，称为理论频率。实际上由随机数发生器产生的 U_i 落在每一个子区间上的个数不可能恰好等于 n，而为 $n_j(j=1,2,\cdots,K)$，这样就会有偏差。频率检验就是检验实际频率与理论频率之间的偏差大小，一般采用 χ^2 检验，即

$$\chi^2=\sum_{j=1}^{K}\frac{(n_j-n)^2}{n} \tag{8.33}$$

显然，若 $n_j=n$，则 χ^2 等于 0，即实际频率与理论频率完全一致，χ^2 的大小反映了偏差程度，也就是随机数的均匀性程度。

随机数发生器 χ^2 检验步骤如下：

(1) 原假设 H_0：给定随机数发生器产生的 U_i 是独立同分布 U(0,1)的随机变量；

(2) 将[0,1]分成 K 个等长的子区间；

(3) 由该随机数发生器产生 N 个随机数 U_i；

(4) 统计计算在每个子区间上的随机数的个数 $n_j(j=1,2,\cdots,K)$；

(5) 计算 $\chi^2=\sum_{j=1}^{K}(n_j-N/K)^2/(N/K)$；

(6) 在原假设条件下，χ^2 接近 $K-1$ 自由度的 χ^2 分布，规定检验水平 α，若

$$\chi^2>\chi^2_{K-1,1-\alpha}$$

则拒绝 H_0，否则不拒绝 H_0。

8.5.2 独立性检验

一个随机数序列可以是均匀分布的，但却不一定满足独立性的要求。检验独立性的常用方法是计算该序列相邻一定间隔的随机数之间的相关系数，然后判断其相关程度。因为

相关系数为 0 是两个随机变量相互独立的必要条件，其值大小可以评价相关程度。

对于给定的随机数发生器，由它产生 N 个随机数 $U_i(i=1,2,\cdots,N)$，则前后相隔为 j 个数的相关系数的均值为

$$\bar{\rho}_j = \left[\frac{1}{N-j}\sum_{i=1}^{N-j}U_iU_{i+j} - \left(\frac{1}{2}\right)^2\right]\Big/S^2 \tag{8.34}$$

其中，S^2 为随机数的方差，即

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(U_i - \frac{1}{2}\right)^2 \tag{8.35}$$

随机数发生器独立性检验步骤如下：

(1) 原假设 H_0：给定随机数发生器产生的 U_i 是独立同分布 U(0,1)的随机变量，即 $\rho=0$；

(2) 由该随机数发生器产生 N 个 U_i，并计算 $\bar{\rho}_j$；

(3) 若 $N-j$ 充分大(一般要求>50)，取统计量

$$\mu = \bar{\rho}_j\sqrt{N-j} \tag{8.36}$$

渐近服从标准正态分布 N(0,1)；

(4) 给定检验水平 α，记 $Z_{1-\alpha}$ 为 N(0,1)上 $1-\alpha$ 的临界点，则当

$$|\mu| > Z_{1-\alpha}, \quad \text{拒绝 } H_0$$

$$|\mu| \leqslant Z_{1-\alpha}, \quad \text{不拒绝 } H_0$$

一般的仿真语言中都提供了若干个随机数发生器，在使用某种仿真语言进行仿真时，应该对其均匀性及独立性进行检验。我们曾对几个仿真语言的随机数进行过检验，发现有些仿真语言提供的发生器的性能是比较差的。例如曾经在国内比较流行的 GPSS-F 仿真语言提供了 30 个随机数发生器，经检验发现，其中有 6 个不能通过均匀性检验，有 2 个不能通过独立性检验。而 SLAM Ⅱ 中提供的 10 个随机数发生器，则都通过了上述两种检验。表 8.13 及表 8.14 中列出了检验结果。

表 8.13 SLMA Ⅱ 随机数发生器($a=0.05, C=0.0$)

编号	$a=65\,539.0$ $m=2\,147\,483\,647.0$		均匀性 $K=10$ $N=10\,000$	独立性 $N=1\,000$	
			χ^2	ρ_1	$\lvert\mu\rvert$
1	Z_0	1 624 217 675.0	4.116	0.009 316	0.294
2		2 014 632 579.0	8.334	0.016 746	0.529
3		2 036 774 231.0	6.086	0.017 945	0.567
4		1 254 240 657.0	10.784	−0.003 828	0.121
5		1 410 143 363.0	8.148	0.012 672	0.401
6		2 135 621 895.0	12.898	0.023 103	0.730
7		1 274 321 477.0	7.252	0.021 203	0.670
8		1 452 313 571.0	7.98	0.014 856	0.470
9		2 135 124 613.0	9.826	0.027 797	0.879
10		1 743 251 541.0	9.154	−0.000 468	0.015

表 8.14 GPSS-F 随机数发生器($a=0.05, Z_0=1.0$)

编号	$a=65\,539.0$ $m=2\,147\,483\,647.0$		均匀性 $K=10$ $N=10\,000$	独立性 $N=1\,000$	
			χ^2	$\bar{\rho}_1$	$\lvert\mu\rvert$
1	C	10 753 813.0	10.550	−0.025 038	0.791
2		228 181 237.0	23.574*	−0.001 900	0.060
3		348 984 893.0	9.078	−0.000 988	0.031
4		590 634 277.0	15.878	−0.016 740	0.529
5		10 841 757.0	7.726	−0.008 008	0.253
6		107 408 213.0	12.060	−0.030 834	0.975
7		469 770 781.0	9.010	0.011 821	0.374
8		832 159 853.0	4.926	−0.019 04	0.602
9		10 842 045.0	13.108	0.013 447	0.425
10		228 244 373.0	8.574	−0.009 017	0.285
11		348 969 437.0	100.354*	0.007 798	0.246
12		10 763 125.0	16.134	0.015 384	0.486
13		107 452 989.0	15.208	0.010 948	0.346
14		469 773 661.0	13.186	−0.011 829	0.974
15		590 648 085.0	25.566*	−0.044 629	1.411
16		10 784 189.0	15.284	0.001 341	0.042
17		228 206 429.0	9.616	0.008 924	0.282
18		469 793 853.0	5.708	0.020 280	0.641
19		832 230 813.0	5.368	0.013 743	0.434
20		107 464 661.0	10.482	−0.015 138	0.478
21		469 808 301.0	61.012*	0.010 780	0.341
22		590 585 901.0	23.294*	0.022 072	0.698
23		228 178 005.0	30.664*	−0.024 098	0.762
24		349 045 949.0	8.592	−0.029 898	0.945
25		10 845 149.0	13.444	0.011 009	0.348
26		107 441 485.0	11.894	−0.004 527	0.143
27		469 795 933.0	10.810	−0.022 571	0.713
28		349 062 285.0	9.968	0.000 600	0.019
29		107 380 645.0	9.572	−0.076 621	2.422**
30		10 767 581.0	15.970	0.069 531	2.198**

注：带 * 者未通过均匀性检验，带 * * 者未通过独立性检验。

由于随机数发生器是离散事件系统仿真的基础，在使用时对其进行均匀性与独立性检验是必不可少的步骤。

8.6 随机变量产生的原理

产生随机变量的方法有许多种，对于给定的随机变量，可根据其特点选择其中一种或几种方法。那么仿真对产生随机变量的方法有何要求呢？

首先是准确性要求，即由这种方法产生的随机变量应准确地具有所要求的分布；其次是快速性要求，在离散事件仿真中，一次运行往往需要产生几万甚至几十万个随机变量，这样，产生随机变量的速度将极大地影响仿真执行的效率。

本节介绍四类最常用的产生随机变量的方法，即反变换法、组合法、卷积法及舍选法。

8.6.1 反变换法

反变换法是最常使用且最直观的方法，它以概率积分变换定理为基础。

设随机变量 x 的分布函数为 $F(x)$；为了得到随机变量的抽样值，先产生在[0,1]区间上均匀分布的独立随机变量 u，由反分布函数 $F^{-1}(u)$ 得到的值即为所需要的随机变量 x

$$x = F^{-1}(u)$$

这种方法是对分布函数进行反变换，因而取名为反变换法。

反变换法的原理可用图 8.11 加以说明。

随机变量概率分布函数 $F(x)$ 的取值范围为[0,1]。现以在[0,1]上均匀分布的独立随机变量作为 $F(x)$ 的取值规律，则落在 Δx 内的样本个数的概率就是 ΔF；从而随机变量 x 在区间 Δx 内出现的概率密度函数的平均值为 $\Delta F/\Delta x$；当 Δx 趋于 0 时，其概率密度函数就等于 $\mathrm{d}F/\mathrm{d}x$，即符合原来给定的密度分布函数，满足正确性要求。

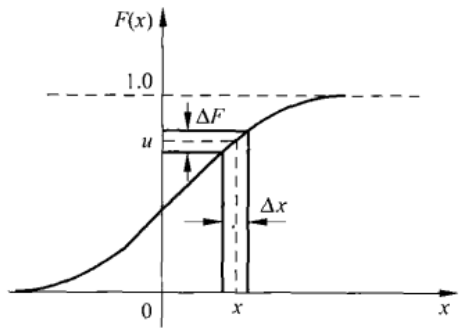


图 8.11 连续分布函数的反变换法原理

例 8.8 设随机变量 x 是[a,b]上均匀分布的随机变量，即

$$f(x) = \begin{cases} \dfrac{1}{b-a}, & a \leqslant x \leqslant b \\ 0, & \text{其他} \end{cases}$$

试用反变换法产生 x。

由 $f(x)$ 可得到 x 的分布函数为

$$F(x) = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \leqslant x \leqslant b \\ 1, & x > b \end{cases} \tag{8.37}$$

用随机数发生器产生 U(0,1) 随机变量 u，并令

$$u = F(x) = \frac{x-a}{b-a} (a \leqslant x \leqslant b)$$

从而可得

$$x = a + (b-a)u$$

■

例 8.9 在第 7 章里曾讨论过指数分布随机变量 x 的产生问题。设 x 的分布函数为

$$F(x) = \begin{cases} 1 - \mathrm{e}^{-x/\beta}, & x \geqslant 0 \\ 0, & \text{其他} \end{cases}$$

试用反变换法产生随机变量 x。

先用随机数发生器产生 $u\sim U(0,1)$，并令

$$u = F(x) = 1 - e^{-x/\beta}$$

从而可得

$$x = F^{-1}(x) = -\beta\ln(1-u)$$

由于 $u\sim U(0,1)$，则 $1-u\sim U(0,1)$，即 u 与 $1-u$ 的分布相同，则上式可改写成

$$x = -\beta\ln u$$

■

由上面两个例子可以看到，用反变换法产生随机变量时首先必须用随机数发生器产生在[0,1]上均匀分布的独立的 u，以此为基础得到的随机变量 x 才能保证分布的正确性。可见，选择一个均匀性及独立性较好的随机数发生器在产生随机变量中的重要地位。

当 x 是离散随机变量时，其反变换法的形式略有不同，原因在于离散随机变量的分布函数也是离散的，因而不能直接利用反函数来获得随机变量的抽样值。下面讨论这类随机变量的反变换法。

设离散随机变量 x 分别以概率 $p(x_1),p(x_2),\cdots,p(x_n)$ 取值 $x_1,x_2,\cdots,x_n$，其中 $0<p(x_i)<1$，且 $\sum_{i=1}^{n} p(x_i)=1$，其分布函数如图 8.12 所示。

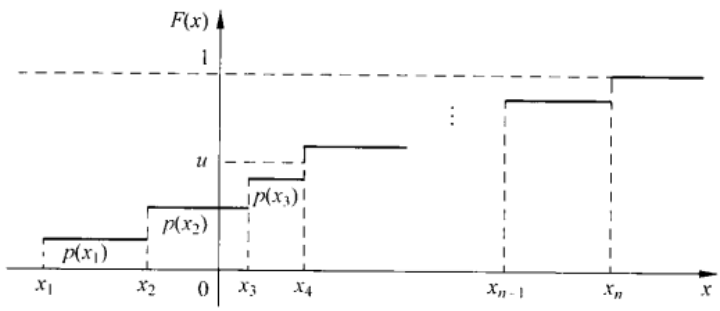


图 8.12 离散分布的反变换法

为使用反变换法获得离散随机变量，先将[0,1]区间按 $p(x_1),p(x_2),\cdots,p(x_n)$ 的值分成 n 个子区间，然后产生在[0,1]区间上均匀分布的独立的随机数 u；根据 u 的值落在何区间，相应区间对应的随机变量就是所需要的随机变量 x_i。

在实际实现时，先要将 x_i 按从小到大的顺序进行排序，即 $x_1<x_2<\cdots<x_n$，从而得到分布函数子区间的分界点 $(0,p(x_1)],(p(x_1),p(x_1)+p(x_2)],\cdots,\left(\sum_{i=1}^{n-1} p(x_i),\sum_{i=1}^{n} p(x_i)\right]$。若由随机数发生器产生的 $u\leqslant p(x_1)$，则令 $x=x_1$，若 $p(x_1)<u\leqslant p(x_1)+p(x_2)$，则令 $x=x_2$，…，依次下去。

例 8.10 设离散随机变量 x 的质量函数及累积分布函数如表 8.15 所示，用反变换法产生随机变量 x。

表 8.15 离散随机变量 x 的质量函数及累积分布函数

x_i	0	1	2	3	4	5
$p(x)$	0	0.1	0.51	0.19	0.15	0.05
$F(x)$	0	0.1	0.61	0.80	0.95	1.00

先由随机数发生器产生[0,1]区间上均匀分布的随机变量 u，设 $u=0.72$，按反变换法，先判断是否 $u\leqslant F(x_1)$，条件不满足，再判断 $u\leqslant F(x_2)$，仍不满足，再判断 $u\leqslant F(x_3)$，满足 $F(x_2)<u\leqslant F(x_3)$，从而得到 $x=x_3=3$。 ■

综上所述，离散随机变量反变换法可描述如下：

(1) 按 x_i 呈递增顺序排列 $p(x_i)$ $(i=0,1,\cdots,N)$。

(2) 产生 $u\sim U(0,1)$。

(3) 求非负整数 I，满足

$$\sum_{j=0}^{I-1}p(x_j)<u\leqslant\sum_{j=0}^{I}p(x_j) \tag{8.38}$$

(4) 令 $x=x_I$。

显然，离散随机变量反变换法的速度主要决定于区间搜索的速度，区间搜索的方法不同，由此产生了各种形式的反变换法。

8.6.2 组合法

反变换法是最直观的方法，但却不一定在任何情况下都是最有效的方法。

当一个分布函数可以表示成若干个其他分布函数之和，而这些分布函数较原来的分布函数更易于取样时，则宜采用本节介绍的组合法。

设随机变量 x 的分布函数 $F(x)$可写成式(8.39)的形式，即

$$F(x)=\sum_{j=1}^{\infty}p_jF_j(x) \tag{8.39}$$

其中，$p_j\geqslant 0$，且 $\sum_{j=0}^{\infty}p_j=1$，$F_j(x)$是其他类型的分布函数；或者随机变量 x 的密度函数 $f(x)$可写成式(8.40)的形式，即

$$f(x)=\sum_{j=1}^{\infty}p_jf_j(x) \tag{8.40}$$

其中，p_j 的定义与前面的相同，$f_j(x)$是这种类型的密度函数，与它相应的分布函数为 $F_j(x)$，则组合法产生随机变量的步骤如下：

(1) 产生一个随机整数 J，满足

$$P\{J=j\}=p_j \quad (j=1,2,\cdots)$$

(2) 产生具有分布函数 $F_j(x)$的随机变量 x_j；

(3) 令 $x=x_j$。

显然，其中第一步是确定采用哪一个分布函数来取样，这可采用离散反变换法来实现。在确定了分布函数后，第二步以该分布函数产生随机变量，如果该分布易于取样，则也易于得到我们所要求的随机变量。具体方法可采用上节介绍的反变换法或者后面将要介绍的其他方法。

例 8.11 设密度函数为 $f(x)=0.5e^{-|x|}$，试产生服从该分布的随机变量 x。

显然，直接用反变换法产生随机变量 x 比较困难，下面我们采用组合法。

分析 $f(x)$的特点(参阅图 8.13),可以看到,它以纵轴为对称轴分为两部分。

我们将 $f(x)$写成如下形式

$$f(x)=0.5\mathrm{e}^{x}I_{(-\infty,0)}(x)+0.5\mathrm{e}^{-x}I_{[0,\infty)}(x)$$

其中

$$I_{(-\infty,0)}(x)=\begin{cases}1, & x\in(-\infty,0)\\ 0, & \text{其他}\end{cases}$$

$$I_{[0,\infty)}(x)=\begin{cases}1, & x\in[0,\infty)\\ 0, & \text{其他}\end{cases}$$

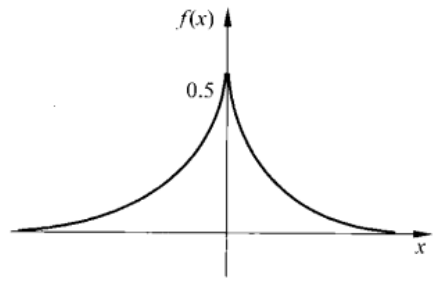


图 8.13 $f(x)=0.5\mathrm{e}^{-|x|}$密度函数图形

令

$$f_1(x)=\mathrm{e}^{x}I_{(-\infty,0)}(x)\quad(p_1=0.5),\qquad f_2(x)=\mathrm{e}^{-x}I_{[0,\infty)}(x)\quad(p_2=0.5)$$

从而

$$f(x)=\sum_{i=1}^{2}p_i f_i(x)$$

■

那么,用组合法产生随机变量 x 的方法如下:

(1) 由 U(0,1)产生随机变量 u_1 及 u_2;

(2) 若 $u_1<0.5$,则由 $f_1(x)$的分布函数产生,即可由反变换法易于得到 x;

$$x=\ln u_2$$

(3) 若 $u_1\geqslant 0.5$,则由 $f_2(x)$的分布函数产生,即 $x=-\ln u_2$。

由上面的例子可以看到,由组合法产生随机变量时,至少要产生两个[0,1]间均匀分布的独立的随机数 U(0,1),而反变换法则只需产生一个 U(0,1)。

8.6.3 卷积法

设随机变量 x 可表示为若干个独立同分布的随机变量 $Y_1,Y_2,\cdots,Y_m$ 之和,即

$$x=Y_1+Y_2+\cdots+Y_m \tag{8.41}$$

则 x 的分布函数与 $\sum_{i=1}^{m}Y_i$ 的分布函数相同,此时称 x 的分布为 Y_i 分布的 m 重卷积。那么,为产生 x,可先独立地从相应分布函数产生随机变量 $Y_1,Y_2,\cdots,Y_m$,然后利用式(8.41)得到 x,这就是卷积法。

例 8.12 试产生均值为 β 的 m 维厄兰分布随机变量 x。

由于均值为 β 的 m 维厄兰分布 Erlang(m,β)的随机变量可表示为 m 个均值为 β/m 的独立的指数随机变量之和,那么可采用卷积法产生 x,其步骤如下:

(1) 独立地产生 m 个 U(0,1)随机数 u_i;

(2) 用反变换法分别产生 Y_i,即

$$Y_i=-\frac{\beta}{m}\ln u_i\quad(i=1,2,\cdots,m)$$

(3) 令

$$x=\sum_{i=1}^{m}Y_i=Y_1+Y_2+\cdots+Y_m$$

■

这较之直接由厄兰分布产生 x 要方便得多。当然，上述方法尚可改进。考虑到对数运算速度较慢，将上述第(2)(3)两步改进为

(2) 计算

$$\prod_{i=1}^{m} u_i = u_1 u_2 \cdots u_m$$

(3) 令

$$x = -\frac{\beta}{m} \ln \prod_{i=1}^{m} u_i$$

这是考虑到乘法运算速度较对数运算要快，因而可提高计算效率。

8.6.4 舍选法

上面介绍的三种方法有一个共同的特点，即直接面向分布函数，因而称为直接法，它以反变换法为基础。

当反变换法难于使用时(例如随机变量的分布函数不存在封闭形式)，舍选法是主要方法之一，下面我们先介绍这种方法的基本思想。

设随机变量 x 的密度函数为 $f(x)$，$f(x)$的最大值为 C，x 的取值范围为[0,1]。

若独立地产生两个[0,1]区间内均匀分布的随机变量 u_1，u_2，则 Cu_1 是在[0,C]区间内均匀分布的随机变量，若以 u_2 求 $f(u_2)$的值，显然，满足

$$Cu_1 \leqslant f(u_2) \tag{8.42}$$

的概率为

$$P\{Cu_1 \leqslant f(u_2)\} = \int_0^1 \mathrm{d}x \int_0^{f(x)} \mathrm{d}y/(1 \times C) = \frac{1}{C}$$

舍选法的做法是：若式(8.42)成立，则选取 u_2 为所需要的随机变量 x，即 $x=u_2$，否则舍弃 u_2。

下面，我们结合图 8.14 来解释舍选法的正确性。从图形上看，在 $1\times C$ 这块矩形面积上任设一点(p_1,p_2)，纵坐标为 Cu_1，横坐标为 u_2，若该点位于 $f(x)$曲线下面，则认为抽样成功。成功的概率为 $f(x)$下的面积除以总面积 C，$f(x)$下的面积的值等于分布函数的值。由于我们假设随机变量 x 的取值范围为[0,1]，因而该面积的值为 1，那么成功的概率就是 $1/C$，成功抽样的点符合所需要的分布。

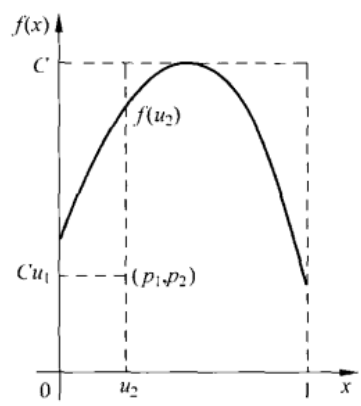


图 8.14 舍选法图示

对一般情形，舍选法则是根据 $f(x)$的特征规定一个函数 $t(x)$(前面实际上是一个常数 C)，对 $t(x)$的要求是：

(1) $t(x) \geqslant f(x)$；

(2) $\int_{-\infty}^{\infty} t(x)\mathrm{d}x = C < \infty$；

(3) 易于从 $t(x)$进行反变换。

这样，令 $r(x)=\frac{1}{C}t(x)$，则 $\int_{-\infty}^{\infty} r(x)\mathrm{d}x = \int_{-\infty}^{\infty} \frac{1}{C}t(x)\mathrm{d}x = 1$。从而可将 $r(x)$看做是一个

密度函数，并用 $r(x)$ 代替 $f(x)$ 取样，以得到所需要的随机变量。

由于 $r(x)$ 并不是要求的 $f(x)$，这就产生选取与舍弃问题，一般算法是：

(1) 产生 $u_1 \sim U(0,1)$；

(2) 由 $r(x)$ 独立地产生随机变量 u_2；

(3) 检验如下不等式

$$u_1 \leqslant f(u_2)/t(u_2) \tag{8.43}$$

若不等式成立，则令 $x=u_2$，否则返回(1)。

例 8.13 随机变量 x 的密度函数为如下 β 分布

$$f(x)=\begin{cases}60x^3(1-x)^2, & 0\leqslant x\leqslant 1\\ 0, & \text{其他}\end{cases}$$

试用舍选法产生服从该分布的随机变量。

显然，若采用反变换法，则必须求解多项式的根，效率很低。采用舍选法首先要选择 $t(x)$，因要求 $t(x)\geqslant f(x)$，故先求 $f(x)$ 极值，可令 $\mathrm{d}f/\mathrm{d}x=0$，得 $x=0.6$，$f(0.6)=2.0736$。$t(x)$ 应是以 x 为自变量的函数，且应易于求反变换，最常采用的是令 $t(x)$ 为一常数，可取

$$t(x)=\begin{cases}2.0736, & 0\leqslant x\leqslant 1\\ 0, & \text{其他}\end{cases}$$

满足 $t(x)\geqslant f(x)$，从而

$$\int_{-\infty}^{\infty} t(x)\mathrm{d}x=\int_0^1 2.0736\mathrm{d}x=2.0736=C$$

$$r(x)=\frac{1}{C}t(x)$$

即

$$r(x)=\begin{cases}1, & 0\leqslant x\leqslant 1\\ 0, & \text{其他}\end{cases}$$

■

所以 $r(x)$ 为[0,1]区间均匀分布的密度函数，从而用舍选法产生 x 的算法如下(参阅图 8.15)：

(1) 产生 $u_1 \sim U(0,1)$；

(2) 由 $r(x)$ 独立地产生 u_2，即 $u_2 \sim U(0,1)$；

(3) 检验 $u_1 \leqslant 60u_2^3(1-u_2)^2/2.0736$，若满足，则令 $x=u_2$，否则返回(1)。

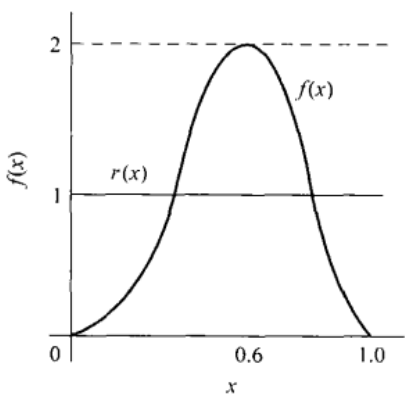


图 8.15 $f(x)=60x^3(1-x)^2$ 舍选法

在使用舍选法时，我们总是希望：①易于从 $r(x)$ 中产生 u_2；②在(3)中舍弃的概率要小。从前面的讨论可知，舍弃的概率为 $1-1/C$，则要求 $\int_{-\infty}^{\infty} t(x)\mathrm{d}(x)=C$ 接近 1。一般同时满足上述两方面要求比较困难。往往是先确定 $r(x)$，$r(x)$ 采用常见的密度函数，如均匀、指数、正态分布等，然后选择 C，使 $t(x)=Cr(x)\geqslant f(x)$，并使 C 尽可能地接近 1。

如果随机变量 x 的取值不在[0,1]区间，而是在[a,b]区间，则可令

$$u_2'=a+u_2(b-a)$$

然后，在舍选判断时用 u_2' 代替 u_2。

对于给定的某种分布，产生随机变量的方法可能有许多种，在 8.7 节中我们介绍几种典型随机变量的产生方法。

8.7 典型随机变量的产生

在 8.6 节中，讨论了产生随机变量的一般方法，并通过举例介绍了均匀分布、指数分布、厄兰分布随机变量的产生。本节介绍其他典型随机变量的产生方法。

8.7.1 连续随机变量的产生

1. 正态分布

先讨论标准正态分布 N(0,1)随机变量的产生。

尽管正态分布随机变量的分布函数没有直接的封闭形式，但若将其转换到极坐标系后，则可以得到其封闭形式。因此，可采用反变换法产生正态分布随机变量，下面对这一方法加以说明。

设 x_1, x_2 是两个独立的 N(0,1)随机变量，则其联合密度函数是

$$f(x_1, x_2) = \frac{1}{2\pi} e^{(x_1^2 + x_2^2)/2}$$

将其转换成极坐标形式

$$\begin{cases} x_1 = \rho\cos\phi \\ x_2 = \rho\sin\phi \end{cases}$$

则

$$f(\rho, \phi) = f(x_1, x_2) \cdot |J|$$

其中 $|J|$ 为雅可比行列式，即

$$|J| = \begin{vmatrix} \dfrac{\partial x_1}{\partial \rho} & \dfrac{\partial x_1}{\partial \phi} \\ \dfrac{\partial x_2}{\partial \rho} & \dfrac{\partial x_2}{\partial \phi} \end{vmatrix} = \begin{vmatrix} \cos\phi & -\rho\sin\phi \\ \sin\phi & \rho\cos\phi \end{vmatrix} = \rho$$

从而可得

$$f(\rho, \phi) = \rho e^{-\frac{\rho^2}{2}} \cdot \frac{1}{2\pi} = f(\rho) \cdot f(\phi)$$

其中 $f(\rho), f(\phi)$ 分别为随机变量 ρ, ϕ 的密度函数

$$\begin{cases} f(\rho) = \rho e^{-\frac{\rho^2}{2}} \\ f(\phi) = \dfrac{1}{2\pi} \end{cases}$$

它们相应的分布函数为

$$\begin{cases} F(\rho) = \int_0^{\rho} \rho' e^{-\frac{\rho^2}{2}} d\rho' = 1 - e^{-\frac{\rho^2}{2}} \\ F(\phi) = \int_0^{\phi} \frac{1}{2\pi} d\phi' = \frac{1}{2\pi}\phi \end{cases}$$

对随机变量 ρ,ϕ 来说，它们的分布函数均具有封闭形式。因而可采用反变换法，即独立地产生两个[0,1]区间上均匀分布的随机数 u_1,u_2，分别对 $F(\rho)$及 $F(\phi)$进行反变换，可得

$$\begin{cases} \phi = 2\pi u_1 \\ \rho = \sqrt{-2\ln(1-u_2)} \end{cases}$$

由于 $1-u_2$ 也是 U(0,1)随机变量，故可令

$$\rho = \sqrt{-2\ln u_2}$$

根据 x_1,x_2 与 ρ,ϕ 之间的变换关系，可得

$$\begin{cases} x_1 = (-2\ln u_2)^{1/2}\cos 2\pi u_1 \\ x_2 = (-2\ln u_2)^{1/2}\sin 2\pi u_2 \end{cases} \tag{8.44}$$

采用上述反变换法，每次产生一对服从正态分布的独立的随机变量。这种方法直观，易于理解，但由于要进行三角函数及对数函数运算，因而计算速度较慢。后来，人们基于舍选法提出了一种效率较高的方法，这种方法的原理说明如下：

(1) 独立产生两个 U(0,1)随机变量 u_1,u_2；

(2) 令 $V_1=2u_1-1,V_2=2u_2-1,W=V_1^2+V_2^2$；

(3) 若 $W>1$，舍弃，返回(1)；

(4) 否则，令

$$\begin{cases} x_1 = V_1[(-2\ln W)/W]^{1/2} \\ x_2 = V_2[(-2\ln W)/W]^{1/2} \end{cases} \tag{8.45}$$

这种方法的舍弃概率为 $1-\frac{\pi}{4}\approx 0.2146$。

如果要求产生 $N(\mu,\sigma^2)$随机变量 x，可先产生 N(0,1)随机变量 x，然后进行如下线性变换即可得到 x'，即

$$x' = \mu + \sigma x$$

正态分布是使用最为广泛的分布之一，原因不仅在于它能代表许多随机变量的分布，而且它是舍选法中作为 $t(x)$的最常选择的分布函数之一。

2. Gama(γ)分布

γ 分布的密度函数为

$$f(x) = \begin{cases} \frac{\beta^{\alpha} x^{\alpha-1} e^{-x/\beta}}{\Gamma(\alpha)}, & x > 0 \\ 0, & \text{其他} \end{cases}$$

其中

$$\Gamma(\alpha) = \int_0^{\infty} t^{\alpha-1} e^{-t} dt$$

在一般情形下，γ 分布的分布函数无封闭形式，难以采用反变换法，只能采用舍选法。

考虑到 γ 分布的如下特性：

若 $x\sim\gamma(\alpha,1)$，则当 $x'=\beta x(\beta>0)$ 时，$x'\sim\gamma(\alpha,\beta)$，所以只需讨论如何产生 $\gamma(\alpha,1)$ 随机变量就行了。

又 $\gamma(1,1)$ 为指数分布 $f(x)=\mathrm{e}^{-x}$，可以直接用反变换法产生 x，因而对 $\gamma(\alpha,1)$ 可分如下两种情况，即 $0<\alpha<1$ 及 $\alpha>1$。

先讨论 $0<\alpha<1$ 时的舍选法，此时可选择 $t(x)$ 为如下形式，即

$$t(x)=\begin{cases}0, & x\leqslant 0\\ \dfrac{x^{\alpha-1}}{\Gamma(\alpha)}, & 0<x\leqslant 1\\ \dfrac{\mathrm{e}^{-x}}{\Gamma(\alpha)}, & x>1\end{cases}$$

可得到

$$C=\int_0^{+\infty}t(x)\mathrm{d}x=\int_0^1\frac{x^{\alpha-1}}{\Gamma(\alpha)}\mathrm{d}x+\int_1^{+\infty}\frac{\mathrm{e}^{-x}}{\Gamma(\alpha)}\mathrm{d}x=b/(\alpha\Gamma(\alpha))$$

其中，$b=(\mathrm{e}+\alpha)/\mathrm{e}$。则密度函数 $r(x)$ 为

$$r(x)=\begin{cases}0, & x\leqslant 0\\ \dfrac{\alpha x^{\alpha-1}}{b}, & 0<x\leqslant 1\\ \dfrac{\alpha\mathrm{e}^{-x}}{b}, & x>1\end{cases}\tag{8.46}$$

$r(x)$ 的相应分布函数具有如下封闭形式

$$R(x)=\begin{cases}\dfrac{x^{\alpha}}{b}, & 0\leqslant x\leqslant 1\\ 1-\dfrac{\alpha\mathrm{e}^{-x}}{b}, & x>1\end{cases}\tag{8.47}$$

这样，可对 $R(x)$ 进行反变换

$$R^{-1}(u)=\begin{cases}(bu)^{1/\alpha}, & u\leqslant\dfrac{1}{b}\\ -\ln\dfrac{b(1-u)}{\alpha}, & \text{其他}\end{cases}$$

因此，当 $0<\alpha<1$ 时，用舍选法产生 $\gamma(\alpha,1)$ 随机变量 x 的算法如下：

(1) 产生 $u_1\sim\mathrm{U}(0,1)$；

(2) 独立产生 $u_2\sim\mathrm{U}(0,1)$，令 $P=bu_2$，若 $P>1$，则去(4)；

(3) 令 $Y=P^{1/\alpha}$，若 $u_1<\mathrm{e}^{-Y}$，则选取，且令 $x=Y$，否则舍弃，返回(1)；

(4) 令 $Y=-\ln[(b-P)/\alpha]$，若 $u_1<Y^{\alpha-1}$，则选取，且令 $x=Y$，否则舍弃，返回(1)。

当 $\alpha>1$ 时，$t(x)$ 的选择方法与 $0<\alpha<1$ 完全不同。设 $\lambda=(2\alpha-1)^{1/2}$，$\mu=\alpha^{\lambda}$，且 $C=4\alpha^{\alpha}\mathrm{e}^{-\alpha}/[\lambda\Gamma(\alpha)]$，则

$$t(x)=Cr(x)$$

其中

$$r(x)=\begin{cases}\dfrac{\lambda\mu x^{\lambda-1}}{(\mu+x^{\lambda})^2}, & x>0\\ 0, & \text{其他}\end{cases}\tag{8.48}$$

$r(x)$相应的分布函数具有封闭形式

$$R(x)=\begin{cases}\dfrac{x^2}{\mu+x^2}, & x>0\\ 0, & \text{其他}\end{cases} \tag{8.49}$$

此时，对 $R(x)$易于进行反变换

$$R^{-1}(u)=\left(\frac{\mu u}{1-u}\right)^{1/\lambda}$$

因此，$\alpha>1$ 时，$\gamma(\alpha,1)$的舍选法步骤如下：

(1) 独立产生 $u_1\sim U(0,1)$，$u_2\sim U(0,1)$；

(2) 令 $V=(2\alpha-1)^{1/2}\ln[u_2/(1-u_2)]$；

(3) 若 $\ln(u_1,u_2^2)<(\alpha-\ln 4+\sqrt{2\alpha-1})V-\alpha e^V$，则选取，令 $x=\alpha e^V$；否则，舍弃，返回(1)。

3. $\beta(\alpha_1,\alpha_2)$分布

$\beta(\alpha_1,\alpha_2)$分布的密度函数为

$$f(x)=\begin{cases}\dfrac{x^{\alpha_1-1}(1-x)^{\alpha_2-1}}{B(\alpha_1,\alpha_2)}, & 0\leqslant x\leqslant 1\\ 0, & \text{其他}\end{cases}$$

其中，$B(\alpha_1,\alpha_2)=\int_0^1 t^{\alpha_1-1}(1-t)^{\alpha_2-1}\mathrm{d}t$。

一般情况下，$\beta(\alpha_1,\alpha_2)$的分布函数也没有封闭形式，亦难以用反变换法产生它的随机变量。但是，我们可以根据该分布的某些特性得到产生这种随机变量的方法。

我们首先讨论在[0,1]区间上的 $\beta(\alpha_1,\alpha_2)$随机变量 x 的产生方法，因为经如下线性变换

$$x'=a+(b-a)x\quad(a<b)$$

则 x'为$[a,b]$区间上的 $\beta(\alpha_1,\alpha_2)$随机变量。

考虑到若 x 是$\beta(\alpha_1,\alpha_2)$随机变量，则 $1-x$ 就是$\beta(\alpha_1,\alpha_2)$随机变量。

对于任意 $\alpha_1>0,\alpha_2>0$，若 $Y_1\sim\gamma(\alpha_1,1)$，$Y_2\sim\gamma(\alpha_2,1)$，且 Y_1 与 Y_2 是独立的，则 $Y_1/(Y_1+Y_2)\sim\beta(\alpha_1,\alpha_2)$，因此，利用前面所述的产生 γ 分布随机变量的方法独立地产生 Y_1,Y_2 后，则易于得到 $\beta(\alpha_1,\alpha_2)$随机变量，其算法可归纳如下：

(1) 分别独立地产生 $Y_1\sim\gamma(\alpha_1,1)$，$Y_2\sim\gamma(\alpha_2,1)$；

(2) 令 $x=Y_1/(Y_1+Y_2)$。

在特殊情况下，如果 $\alpha_1=1$ 或 $\alpha_2=1$，则 $\beta(\alpha_1,1)$或 $\beta(1,\alpha_2)$的密度函数比较简单，其相应的分布函数具有封闭的形式，因而可以用反变换法。例如，若 $\alpha_2=1$，则 $\beta(\alpha_1,1)$的密度函数为

$$f(x)=\begin{cases}\dfrac{x^{\alpha_1-1}}{B(\alpha_1,1)}, & 0\leqslant x\leqslant 1\\ 0, & \text{其他}\end{cases}$$

其中，$B(\alpha_1,1)=\int_0^1 t^{\alpha_1-1}\mathrm{d}t=\dfrac{1}{\alpha_1}$，则 $f(x)=\alpha_1 x^{\alpha_1-1}$，从而其分布函数为

$$F(x)=\begin{cases}x^{a_1}, & 0\leqslant x\leqslant 1\\ 0, & \text{其他}\end{cases}$$

利用反变换法可得

$$x=u^{1/a_1}$$

4. 对数正态分布

对数正态分布的分布函数也无封闭形式，但由对数正态分布与正态分布随机变量之间的关系，可间接地产生这种随机变量。

若 Y 是具有均值 μ 与方差 σ^2 的正态分布随机变量，$Y\sim N(\mu,\sigma^2)$，则 $x=e^Y$ 是对数正态分布的随机变量，其均值与方差分别为

$$\begin{cases}\mu_x=e^{\mu+\sigma^2/2}\\ \sigma_x^2=e^{2\mu+\sigma^2}(e^{\sigma^2}-1)\end{cases}\tag{8.50}$$

那么若采用前面讨论过的反变换法产生正态分布随机变量，则对数正态分布的随机变量可由下面算法产生：

(1) 独立产生 $u_1\sim U(0,1)$，$u_2\sim U(0,1)$；

(2) 令

$$Y_1=\mu+(-2\sigma^2\ln u_1)^{1/2}\cos 2\pi u_2$$
$$Y_2=\mu+(-\sigma^2\ln u_1)^{1/2}\sin 2\pi u_2$$

(3) 令

$$x_1=e^{Y_1},x_2=e^{Y_2}$$

注意，若已知的是对数正态分布的均值与方差，则应利用式(8.50)求出相应正态分布的均值与方差，这样才能使用本方法。

8.7.2 离散随机变量的产生

产生离散随机变量的最基本方法是反变换法。从 8.6.1 节中我们已经看到，离散反变换法要求按 x_i 递增顺序对 $p(x_i)$ 加以排序，然后进行区间搜索，下面就常见的几类离散随机变量的产生方法进行讨论。

1. 离散均匀分布 DU(i,j)

离散均匀分布随机变量 x 取值为[i,j]区间上的整数值，其质量函数及分布函数如下

$$p(x)=\begin{cases}\dfrac{1}{j-i+1}, & x\in\{i,i+1,\cdots,j\}\\ 0, & \text{其他}\end{cases}\tag{8.51}$$

$$F(x)=\begin{cases}0, & x<i\\ \dfrac{\lfloor x\rfloor-i+1}{j-i+1}, & i\leqslant x\leqslant j\\ 1, & x>j\end{cases}\tag{8.52}$$

其中，$\lfloor x \rfloor$表示小于或等于 x 的最大整数。产生这种分布的随机变量方法很简单：

(1) 产生 $u \sim U(0,1)$；

(2) 令 $x=i+\lfloor (j-i+1)u \rfloor$。

2. 几何分布

下面介绍三种产生几何分布随机变量的方法。

(1) 对几何分布的分布函数的反变换法

由其分布函数

$$F(x) = 1-(1-p)^{\lfloor x+1 \rfloor} \tag{8.53}$$

对 $F(x)$进行反变换：

① 产生 $u \sim U(0,1)$；

② 令 $x=\lfloor \ln(1-u)/\ln(1-p) \rfloor$。

这种方法简单直观，缺点是当 p 接近 0 时，$\ln(1-p)$也接近 0；若 p 接近 1，则 $\ln(1-p)$是一个很大的负数。两种情况下，精度不易得到保证，一般要采用双精度运算。

(2) 面向质量函数的反变换法

由几何分布的质量函数

$$p(x) = \begin{cases} p^x(1-p), & x \in \{0,1,2,\cdots\} \\ 0, & \text{其他} \end{cases} \tag{8.54}$$

不难看出，当

$$\begin{aligned}
&x=0, && p(x)=p(0)=1-p \\
&x=1, && p(x)=p(1)=p(1-p)=p\cdot p(0) \\
&\cdots \\
&x=i-1, && p(x)=p(i-1)=p^{i-1}(1-p)=p\cdot p(i-2) \\
&x=i, && p(x)=p(i)=p^{i}(1-p)=p\cdot p(i-1) \\
&\cdots
\end{aligned}$$

从而可以得到这种随机变量的递推算法如下：

① 令 $i=0, p_i=1-p, F_i=0$；

② 产生 $u_{i+1} \sim U(0,1)$；

③ 令 $p_{i+1}=p\cdot p_i, F_{i+1}=F_i+p_{i+1}$；

④ 若 $F_i \leqslant u_{i+1} < F_{i+1}$，则 $x=i+1$，否则 $i=i+1$，并返回③。

(3) 基于几何分布的定义

根据几何分布的定义，在独立的伯努里试验序列中，每次试验成功的概率为 p，则第一次成功之前失败的次数为几何分布；那么，设 $Y_1, Y_2, \cdots$是独立进行伯努里试验的随机变量序列，且令 $x=\min\{i: Y=1\}-1$，则 x 服从几何分布，这样，可得到产生 x 的算法为：

① 令 $i=0$；

② 独立产生 $u \sim U(0,1)$；

③ 若 $u \leqslant p$，则 $x=i$，否则 $i=i+1$，返回②。

3. 二项分布

设每次伯努里试验成功的概率为 p，在 t 次独立的试验中成功的总次数服从二项分布，其质量及分布函数为

$$p(x)=\begin{cases}\binom{t}{x}p^x(1-p)^{t-x}, & x\in\{0,1,\cdots,t\}\\ 0, & \text{其他}\end{cases} \tag{8.55}$$

$$F(x)=\begin{cases}0, & x<0\\ \sum_{i=1}^{\lfloor x\rfloor}\binom{t}{i}p^i(1-p)^{t-i}, & 0\leqslant x\leqslant t\\ 1, & \text{其他}\end{cases} \tag{8.56}$$

其中，$\binom{t}{x}=t!/[x!(t-x)!]$。

显然，直接对 $F(x)$ 进行反变换比较麻烦，我们可以由二项分布的定义找到该分布与伯努里分布的关系，即可用卷积法产生该分布的随机变量：

(1) 独立产生 t 个伯努里随机变量 $Y_1,Y_2,\cdots,Y_t$；

(2) 令 $x=\sum_{i=1}^{t}Y_i$。

而产生伯努里随机变量的算法是：

(1) 独立产生 $u\sim U(0,1)$；

(2) 若 $u\leqslant p$，则 $Y=1$，否则 $Y=0$。

4. 泊松分布

泊松分布的密度函数为

$$p(x)=\begin{cases}\dfrac{e^{-\lambda}\lambda^x}{x!}, & x\in\{0,1,\cdots\}\\ 0, & \text{其他}\end{cases} \tag{8.57}$$

分析 $p(x)$ 的特点，不难看到

$$p(i)=\frac{e^{-\lambda}\lambda^i}{i!}=\frac{\lambda}{i}\frac{e^{-\lambda}\lambda^{i-1}}{(i-1)!}=\frac{\lambda}{i}p(i-1) \tag{8.58}$$

则其分布函数 F_i 可表示成

$$F_i=F_{i-1}+p(i) \tag{8.59}$$

从而可得到产生泊松随机变量的算法如下：

(1) 令 $i=0,p_i=e^{-\lambda},F_i=0,F_{i+1}=p_i$；

(2) 产生 $u_{i+1}\sim U(0,1)$；

(3) 令 $p_{i+1}=\dfrac{\lambda}{i+1}p_i,F_{i+1}=F_i+p_{i+1}$；

(4) 若 $F_i\leqslant u_{i+1}<F_{i+1}$，则 $x=i+1$，否则，$i=i+1$，返回(3)。

8.8 小结

由观测数据确定随机变量模型分为三种情况：已知随机变量分布类型确定其参数，确定随机变量概率分布类型及其参数，理论分布难得到时确定实验分布。由观测数据假设了其分布的类型并估计出其参数以后，一般需要进行拟合优良度检验，本章讨论了两种拟合优良度检验方法。

仿真中采用的随机数发生器不是在概率论意义下的真正的随机数，是伪随机数，但是如果算法选择得合适，由这种算法得到的数据，通过统计检验后能具有较好的统计特性。本章介绍了线性同余发生器和组合发生器，同时，讨论了随机数发生器均匀性和独立性检验。

仿真对产生随机变量的方法有准确性和快速性两个要求，本章着重讨论了反变换法、组合法、卷积法及舍选法共四类最常用的产生随机变量的一般方法，介绍了正态分布、γ 分布、β 分布、对数正态分布共四种典型连续随机变量的产生，以及离散均匀分布、几何分布、二项分布、泊松分布共四种离散随机变量的产生。

习题

1. 给定一个从 -3 到 4 范围内的连续随机变量的概率分布函数如下，开发一个该变量的产生器。

$$F(x)=\begin{cases}0, & x\leqslant -3\\ \dfrac{1}{2}+\dfrac{x}{6}, & -3<x\leqslant 0\\ \dfrac{1}{2}+\dfrac{x^2}{32}, & 0<x\leqslant 4\\ 1, & x\geqslant 4\end{cases}$$

2. 一个部件退出服务的时间服从 0 到 8 小时的均匀分布。两个这样的部件串联放入一个系统，当其中一个部件退出服务时，整个系统停止。如果用 X_i 表示部件的运行时间，用 $Y=\min(X_1,X_2)$ 表示系统的生存期。设计两种产生 Y 的不同方法。

3. 利用 K-S 检验法检验下列样本是否符合均值为 0.0、方差为 2.5 的正态分布。检验水平为 $\alpha=0.05$。

1.549 422	2.444 344	−1.356 287	−1.158 468	1.986 288
−1.317 650	1.203 433	−2.405 187	−0.983 101	−0.942 457
2.627 202	2.295 194	0.253 501	0.256 372	−1.221 426
−2.819 277	2.729 291	1.374 238	−0.028 606	0.940 219
−1.100 076	−2.032 944	−1.105 679	1.694 956	0.019 935

4. 若有一批样本数为50的三极管，其放大倍数β值分别为

34.7	56.2	38.4	54.1	57.4	51.7	60.6	67.7	78.1	38.2
49.2	42.8	45.2	53.4	80.4	97.4	84.5	65.3	66.4	73.4
61.1	68.4	69.4	81.3	74.4	36.3	47.2	52.4	69.2	89.7
76.6	67.3	66.2	59.8	59.2	63.2	38.4	44.6	70.1	28.1
52.3	44.5	46.4	64.4	66.4	54.2	78.8	62.0	32.4	48.5

设β为随机变量X，X的取值范围为(0,100)，试计算其均值和方差，且用直方图法确定该随机变量的分布类型，并对其进行$\alpha=0.10$水平的χ^2检验。

5. 试编程分别实现符合下列分布的随机数发生器：

正态分布：均值=3.5，方差=5.8

γ分布：① $\alpha=0.3,\beta=4.0$

② $\alpha=3.3,\beta=4.0$

泊松分布：均值=4.2

6. 某机修厂修理某种机床所需时间的统计数据如下：

修理时间区间(x)	频数	相对频率	累计频率
$0\leqslant x\leqslant 0.5$	31	0.31	0.31
$0.5<x\leqslant 1.0$	10	0.10	0.40
$1.0<x\leqslant 1.5$	25	0.25	0.66
$1.5<x\leqslant 2.0$	34	0.34	1.00

试用连续实验分布函数产生随机变量x。

7. 使用电子数据表产生1 000个指数分布的随机数值，均值为10。仿真值的最大值是多少？所产生的值有多大比例小于平均值10？画出所产生的值的直方图。（提示：如果在你的电子数据表中找不到指数分布随机数的发生器，那么使用公式$-10\times LOG(1-R)$，R是从0到1均匀分布的随机数。LOG是自然对数。）

8. 分别在如下条件下，分析乘同余发生器：

① $a=11,m=16,X_0=7$　　② $a=11,m=16,X_0=8$

③ $a=7,m=16,X_0=7$　　④ $a=7,m=16;X_0=8$

分别针对每种情况产生足够多的数值以完成一个周期。可以得出什么结论？达到最大周期没有？

9. 分析下面这些线性同余法能否达到最大周期，并给出达到最大周期的X_0的限制：

① 混合同余发生器，其中

$a=2\ 814\ 749\ 767\ 109$

$c=59\ 482\ 661\ 568\ 307$

$m=2^{48}$

② 乘同余发生器，其中

$a=69\ 069$

$c=0$

$m=2^{32}$

③ 混合同余发生器，其中

$a=4\ 951$

$c=247$

$m=256$

④ 乘同余发生器，其中

$a=6\ 507$

$c=0$

$m=1\ 024$

10. 对一个 16 位的计算机，L'Ecuyer[1998]建议将如下三个乘同余发生器组合起来：$m_1=32\,363, a_1=157, m_2=31\,727, a_2=146, m_3=31\,657, a_3=142$。组合后的发生器的周期接近 8×10^{12}。使用这个组合发生器，产生 5 个随机数；对于其中每个发生器，分别取初始种子 $X_{i,0}=100\,300\,500(i=1,2,3)$。使用本章中给出的测试方法，对上面习题中的发生器进行测试。

参考文献

[1] Fishman G S. Principles of Discrete Event Simulation. New York: Wiley, 1978

[2] Lehmer D H. Mathematical Methods in Large-Scale Computing: Units. Ann. Computing Lab (Harvard Univ.), 1951, 26: 141-146

[3] Knuth D E. The Art of Computer Programming. Mass.: Addison-Wesley, 1969

[4] Marsaglia G and Bray T A. One-Line Random Number Generators and Their Use in Combinations. Communication Ass. Computing, 1968, 3(11): 757-759

[5] 韩慧君. 系统仿真. 北京：国防工业出版社，1985

[6] Von Neumann J. Various Techniques Used in Connection with Random Digits. Nat. Bur. Std. Math. Ser., 1951, 12: 36-38

[7] Cheng R C H. Generating Beta Variates with Nonintegral Shape Parameters. Communication Ass. Computing, 1978, 3

[8] Cheng R C H. The Generation of Gamma Variables with Nonintegrai Shape Parameter. Appl. Statist., 1977, 26

[9] Kevin Watkins. Discrete Event Simulation in C. McGraw-Hill Book Company, 1990

[10] 卞国瑞等. 概率论. 北京：人民教育出版社，1979

[11] 卞国瑞等. 数理统计. 北京：人民教育出版社，1979

[12] Law A M and Kelton W D. Simulation Modeling and Analysis. McGrawHill, 1982

[13] 熊光楞等. 连续系统仿真与离散事件系统仿真. 北京：清华大学出版社，1991

第 9 章
离散事件系统仿真策略与仿真语言

和连续系统仿真一样，离散事件系统仿真的核心问题是建立描述系统行为的仿真模型。虽然离散事件系统大多是随机系统。但由于仿真模型中采用的是伪随机数，从理论上讲，其状态的转移也是确定的，因而也可得到确定性的状态转移函数。与连续系统的区别在于，离散事件系统的模型难以采用某种规范的形式，而一般采用流程图或网络图的形式才能准确地定义实体在系统中的活动。因此，离散事件系统仿真研究的另一重点是仿真策略问题。

由于离散事件系统的复杂性，仿真技术作为这类系统的分析工具使用日益广泛。采用一般的高级语言（如 FORTRAN、C）实现仿真程序存在着程序量大，人机界面差、不易维护等缺点，因此人们迫切需要功能日益完善的仿真语言作为这类系统研究的支撑工具，以便使用户描述模型更为方便与直观，仿真实验运行更为灵活，仿真结果更易于理解。

本章首先在 9.1 节中分别讨论事件调度（ES）法、活动扫描（AS）法、进程交互（PI）法和三阶段法四种策略的机制，并比较其优缺点；然后在 9.2 节中，介绍目前流行的几种仿真语言仿真建模的特点。

9.1 离散事件系统仿真策略

9.1.1 主要术语

在第 7 章中我们曾经介绍过描述离散事件系统的一些术语。如实体、活动、进程、属性、事件等。正如 Zelgler 所指出的，模型作为系统的规范（system specification），两者之间既有相联系的一方面，也有其区别的一方面。模型是系统的抽象，因而在模型描述中所使用的术语不可能与系统描述完全相同。在进行各种仿真策略的讨论之前，我们先就某些术语进行概念性地说明，并对各种策略中所使用的符号加以定义。

模型的非形式描述中采用如下一些主要术语：

(1) 成分(component)：相应于系统中的实体，用于构造模型中的各个部分，根据其在模型中的作用，可分为两大类：

主动成分(active-type component)：可以主动产生活动的成分。如排队系统中的顾客，它的到达将产生排队活动或服务活动。

被动成分(passive-type component)：本身不能激发活动，只有在主动成分作用下才产生状态变化。

(2) 描述变量：是成分的状态、属性的描述。例如在排队系统中，顾客的到达时间是一个描述变量，它是成分——顾客的一个属性；服务台服务是一个描述变量，它描述了成分——服务台的状态(闲或忙)。

(3) 成分间的相互关系：描述成分之间相互影响的规则。例如在排队系统中，顾客这个成分与服务台这个成分之间，顾客是主动成分，服务台是被动成分。服务台的状态受顾客的影响(作用)，作用的规则是：如果服务台“闲”，顾客的到达则改变其当前状态，使其由“闲”到“忙”；如果服务台“忙”，则对服务台不起作用，而作用到自身——顾客进入排队状态或取消到达。实际上，在一个模型中，主动成分对被动成分可能产生作用，而主动成分之间也可能产生作用。

一个系统的非形式模型描述包括定义成分，描述变量及成分间的相互关系。在本章中采用如下一些符号：

$C=\{\alpha_1,\alpha_2,\cdots,\alpha_n\}$成分集合，$\alpha_i$ 是第 i 个成分分量($1\leqslant i\leqslant n$)。

$C_A=\{\alpha_1,\alpha_2,\cdots,\alpha_m\}$主动成分子集，$\alpha_j$ 是第 j 个主动成分分量($1\leqslant j\leqslant m, m<n$)。

$C_P=\{\alpha_1,\alpha_2,\cdots,\alpha_l\}$被动成分子集，$\alpha_k$ 是第 k 个被动成分分量($1\leqslant k\leqslant l, l<n$)。

显然，在一个模型中，$n=m+l$。

S 所有成分的状态变量，值域为 S_α。

$P=\{p_1,p_2,\cdots,p_n\}$参数(属性)集合。

t_α 成分 α 的状态下一发生变化的时刻，值域为$\{R^{l}_{(0,\infty)}\}$。

$D_\alpha(S)$成分 α 在状态变量值为 S 时的条件是否满足，当 $D_\alpha(S)=true$ 时，表示满足，$D_\alpha(S)=false$ 时表示不满足。

$TIME$ 模型仿真钟的值，值域为$\{R^{+}_{(0,\infty)}\}$。

上述符号以后不加说明地进行引用，其他一些局部使用的符号将分别在有关的部分加以说明。

9.1.2 事件调度法(event scheduling)

正如在第 7 章中讨论过的那样，离散事件系统中最基本的概念是事件。事件的发生引起系统状态的变化。用事件的观点来分析真实系统，通过定义事件及每个事件发生引起系统状态的变化，按时间顺序确定并执行每个事件发生时有关的逻辑关系，这就是事件调度法的基本思想。

按这种策略建立模型时，所有事件均放在事件表中。模型中设有一个时间控制成分，该成分从事件表中选择具有最早发生时间的事件，并将仿真钟修改到该事件发生的时间，再调

用与该事件相应的事件处理模块，该事件处理完后返回时间控制成分。这样，事件的选择与处理不断地进行，直到仿真终止的条件或程序事件产生为止。用这种策略建立仿真模型的方法在第 7 章中已经讨论过，下面给出这种策略的非形式描述。

成分集合 $C=\{\alpha_1,\alpha_2,\cdots,\alpha_n\}$

主动成分集 $C_A=\{\alpha_1,\alpha_2,\cdots,\alpha_m\}$

被动成分集 $C_P=\{\alpha_{m+1},\alpha_{m+2},\cdots,\alpha_{m+n}\}$

描述变量

描述每一主动成分 $\alpha\in C_A$ 的变量 α 的状态 s_α 值域 S_α，s_α 下一变化时刻的时间变量 t_α。

描述 $\alpha\in C_P$ 的变量 α 的状态 s_α，值域 S_α（被动成分的状态变化只有在主动成分作用下才能发生，其发生时间由主动成分来确定，因而不需要时间变量）。

描述所有成分的属性的变量参数集合 $P=\{p_1,p_2,\cdots,p_r\}$ 成分间的相互关系。

在每个主动成分 $\alpha\in C_A$ 的作用下其状态变化的描述，称为事件处理流程；各成分处理的优先级即同时发生时的处理顺序，称为解结规则。在事件调度法中，一般将主动成分也同时定义为被动成分，以便接受其他主动成分的作用。

根据上述非形式描述，事件调度法算法如下：

执行初始化操作，包括

置初始时间 $t=t_0$，结束时间 $t_\infty=t_e$；

事件表初始化，设置系统初始事件；

成分状态初始化 $S=((s_{\alpha_1},t_{\alpha_1}),\cdots,(s_{\alpha_m},t_{\alpha_m}),(s_{\alpha_{m+1}},t_{\alpha_{m+1}}),\cdots,(s_{\alpha_n},t_{\alpha_n}))$

操作事件表，包括：

取出具有 $t(s)=\min\{t_\alpha\mid\alpha\in C_A\}$ 事件记录；

修改事件表；

推进仿真钟　$TIME=t(s)$

While($TIME\leqslant t_\infty$)则执行

　Case　根据事件类型 i

　　$i=1$ 执行第 1 类事件处理程序①

　　$i=2$ 执行第 2 类事件处理程序

　　…

　　$i=m$ 执行第 m 类事件处理程序

　endcase

取出具有 $t(s)=\min\{t_\alpha\mid\alpha\in C_A\}$ 事件记录②

置仿真时间 $TIME=t(s)$

endwhile③

① 第 i 类事件处理程序对成分的状态变化进行建模，而且要进行统计计算。

② 若具有 $t(s)=\min\{t_\alpha\mid\alpha\in C_A\}$ 事件记录有若干个，则按解结规则处理。

③ 该算法中未包括仿真结束后对结果的分析等内容，以下同。

9.1.3 活动扫描法(activity scanning)

事件调度法中仿真钟的推进仅仅依据 $t(s)=\min\{t_\alpha \mid \alpha\in C_A\}$ 准则,而该事件发生的任何条件的测试则必须在该事件处理程序内部去处理。如果条件满足,该事件发生,而如果条件不满足的话,则推迟或取消该事件发生,因此,从本质上来说,事件调度法是一种"预定事件发生时间"的策略。这样,仿真模型中必须预定系统中最先发生的事件,以便启动仿真进程。在每一类事件处理子程序中,除了要修改系统的有关状态外,还要预定本类事件的下一事件将要发生的时间,这一点我们在第 7 章的例子中已经看到过。该策略对于活动持续时间确定性较强(可以是服从某种分布的随机变量)的系统是比较方便的。

但是,当事件的发生不仅与时间有关,而且与其他条件有关,即只有满足某些条件时才会发生。在这种情况下,事件调度法策略的弱点则表现出来了,原因在于,由于这类系统的活动持续时间的不确定性,因而无法预定活动的开始或终止时间。

活动扫描法是针对具有上述特点的系统产生的。这种策略的基本思想是:系统由成分组成,而成分包含着活动,这些活动的发生必须满足某些条件;每一个主动成分均有一个相应的活动子例程;仿真过程中,活动的发生时间也作为条件之一,而且是较之其他条件具有更高的优先权。

设 $D_\alpha(S)$ 表示成分 α 在系统状态 S 下的条件是否满足($D_\alpha(S)=true$ 表示满足,$D_\alpha(S)=false$ 则表示不满足),t_α 表示成分 α 的状态下一发生变化的时刻,活动扫描法每一步要对系统中所有主动成分进行扫描,当①$t_\alpha \leqslant$ 仿真钟当前值 $TIME$,且②$D_\alpha(S)=true$ 时,执行该成分 α 的活动子例程。所有主动成分扫描一遍后,则又按同样顺序继续进行扫描,直到仿真结束。显然,活动扫描法由于包括了对事件发生时间的扫描,因而它也具有事件调度法的功能。

具体实现时,活动扫描法采取以下措施:

(1) 设置系统仿真钟 $TIME$ 与成分仿真钟 t_α。

系统仿真钟表示系统的仿真进程的推进时间,而成分仿真钟则记录该成分的活动发生时刻,两者的关系可能有三种情况:

① $t_\alpha > TIME$ 表示该活动在将来某一时刻可能发生;

② $t_\alpha = TIME$ 表示该活动如果条件满足则应立即发生;

③ $t_\alpha < TIME$ 表示该活动按预定时间早应发生,但因条件未满足,到目前为止实际上仍未发生,当前是否发生,则只要判断其发生的条件。

(2) 设置条件处理模块

该模块用于测定 $D_\alpha(S)$ 的值及系统仿真钟与成分仿真钟之间的关系,记

$$FUTURE(S)=\{\alpha \mid t_\alpha > TIME\},\ PRESENT(S)=\{\alpha \mid t_\alpha = TIME\},$$
$$PAST(S)=\{\alpha \mid t_\alpha < TIME\}$$

该模块将满足下列条件:

① $\alpha\in PRESENT(S)\cup PAST(S)$,且

② $D_\alpha(S)=true$

的成分置于可激活(ACTIVATABLE)的成分集合中,即

$$ACTIVATABLE(S)=\left\{\alpha\middle|\begin{matrix}\alpha\in PRESENT(S)\cup PAST(S)\\ D_\alpha(S)=true\end{matrix}\right.$$

此时，系统仿真钟不推进，仅仅处理成分活动，包括修改成分仿真钟。

如果可激活的成分集合为空，则将系统仿真钟推进到下一最早发生的活动生成时刻，即 $TIME=\min(t_\alpha\in FUTURE(S))$。

活动扫描法的仿真模型中，成分、变量的定义与事件调度法相同，成分间的相互关系除了定义成分的活动外，还包括 $D_\alpha(S)$的定义及解结规则的定义。

下面给出活动扫描法的一种典型算法，以便读者对这种策略有一个总体认识。

执行初始化操作，包括：

设置初始时间 $t=t_0$，结束时间 $t_\infty=t_e$；

设置主动成分的仿真钟：$t_\alpha(i)(i=1,2,\cdots,m)$；

成分状态初始化：$S=((s_{\alpha_1},t_{\alpha_1}),\cdots,(s_{\alpha_m},t_{\alpha_m}),(s_{\alpha_{m+1}},t_{\alpha_{m+1}}),\cdots,(s_{\alpha_n},t_{\alpha_n}))$；

设置系统仿真钟：$TIME=t_0$。

While($TIME<t_\infty$)，则执行扫描

for j=最高优先数到最低优先数

将优先数为 j 的成分置成 i

if($t_\alpha(i)<TIME(i)$且 $D_{\alpha_j}(S)=true$)

执行活动子例程 i

退出，重新开始扫描

endif

endfor

$TIME=\min(t_\alpha\mid\alpha\in FUTURE(S))$

endwhile

从上面的算法中可以看到，以活动的观点建模的活动扫描法其核心是建立活动子例程模型，包括此活动发生引起的状态变化（自身的），对其他成分的状态产生的作用等，而条件处理模块则是这种策略实现的本质，它相应于事件调度法中的定时模块。

9.1.4 进程交互法（process interactive）

离散事件系统仿真建模的第三种方法是进程交互法。从第7章的讨论中可以知道，一个进程包含若干个有序事件及有序活动。采用这种策略建模更接近于实际系统，从用户的观点来看这种策略更易于使用，因而得到迅速地发展。但从这种策略的软件实现来看，较之事件调度法及活动扫描法则要复杂得多。本节就这种策略的软件实现进行讨论。目前流行的许多仿真语言中都具有进程交互法建模的功能，但其软件实现的方法却不尽相同，这里仅以目前使用最为广泛的 GPSS 仿真语言为例来加以讨论。

进程交互法采用进程（process）描述系统，它将模型中的主动成分所发生的事件及活动按时间顺序进行组合，从而形成进程表，一个成分一旦进入进程，它将完成该进程的全部活动。

软件实现时，系统仿真钟的控制程序采用两张事件表：其一是当前事件表（current

events list,CEL),它包含了从当前时间点开始有资格执行的事件的事件记录,但是该事件是否发生的条件(如果有的话)尚未判断;其二是将来事件表(future events list,FEL),它包含在将来某个仿真时刻发生的事件的事件记录。每一个事件记录中包括该事件的若干属性,其中必有一个属性,说明该事件在进程中所处位置的指针。

当仿真钟推进时,满足 $t_a \leqslant TIME$ 的所有事件记录从 FEL 移到 CEL 中,然后对 CEL 中的每个事件记录进行扫描,对于从 CEL 中取出的每一个事件记录,首先判断它属于哪一个进程以及它在该进程中的位置。该事件是否发生则决定于发生条件是否为真。若 $D_{a_i}(S)=True$,则发生包含该事件的活动,只要条件允许,该进程要尽可能多地连续推进,直到结束;如果 $D_{a_i}(S)=False$ 或仿真钟要求停止,则退出该进程,然后对 CEL 的下一事件记录进行处理。当 CEL 中的所有记录处理完毕后,结束对 CEL 的扫描,继续推进仿真钟,即把将来事件表中的最早发生的事件记录移到 CEL 中,直到仿真结束。这种策略可用以下算法来描述:

执行初始化操作,包括
设置初始时间 $t=t_0$,结束时间 t_∞;
设置初始化事件,并置于 FEL 中;
将 FEL 中有关事件记录置于 CEL 中;
成分状态初始化 $S=((s_{a_1},t_{a_1}),\cdots,(s_{a_m},t_{a_m}),(s_{a_{m+1}},t_{a_{m+1}}),\cdots,(s_{a_n},t_{a_n}))$;
设置系统仿真钟 $TIME=t_0$。
While ($TIME \leqslant t_\infty$),则执行。
(1) CEL 扫描
While (CEL 中最后一个记录未处理完) 则
 While ($D_{a_i}(S)=true$ 且当前成分未处理完) 则
 执行该成分的活动①
 确定该成分的下一事件②
 endwhile
endwhile
(2) 推进仿真钟
$TIME$=FEL 中安排的最早时间
 If($TIME \leqslant t_\infty$)则
 将 FEL 中所在 $TIME$ 时刻发生的事件记录移到 CEL 中
 endif
endwhile

由上面的讨论可以看到,进程交互法既可预定事件,又可对条件求值,因而它兼有事件调度法及活动扫描法两者的优点。

① 一个进程可能有若干个活动,每一个活动有相应的活动子例程来处理该成分在此活动发生时的状态变化及对其他成分的作用。

② 为确定该进程的下一事件,必须对 $D_{a_i}(S)$ 求值,以便决定下一活动是否发生。如果条件不满足,则该进程退出,并记下断点,置于 FEL 中。

9.1.5 三阶段法

三阶段法也结合了事件调度法和活动扫描法的特点，将整个仿真控制过程分为三个阶段，程序实现也颇为简练，因而在仿真应用中被大量采纳。本节就三阶段法进行详细讨论。三阶段法中的一些基本概念：

1) B_s 和 C_s

B_s：B类活动，可明确预知起始和结束时间的活动。B源于英文 bound，意在该活动将在界定时间范围内发生。

C_s：C类活动，非B类活动即为C类活动。C源于英文 condition，意在该类活动的发生或结束是有条件的，其发生时间是不可事先预知的。

2) 实体的三个重要属性

三阶段法中每个实体必备的三个属性是：

(1) 时间片(time cell)：下一状态转移时间。只有该实体属于将来某时刻发生的B活动时，该属性才有意义。

(2) 可用性(availability)：是一个取布尔值的标志，用来表示该实体是否能属于将来某时刻发生的B活动，换言之，即将来某时刻发生B活动时，该实体是否可以被无条件占用。如果标志为“真”(*true*)，则说明可用，标志为“假”(*false*)，则说明不可用。

(3) 下一活动(next activity)：像“时间片”属性一样，仅当“可用性”属性为“假”时，该属性才有意义，它表示“时间片”所预期的B活动。

3) 三阶段法的流程图(图 9.1)

(1) A 阶段：时间扫描

扫描事件表，找出下一最早发生事件。将系统仿真钟推进到该事件的发生时刻。系统时钟一直保持这一时刻直到下一个 A 阶段发生。具体做法是：仿真控制程序搜寻出那些“可用性”属性为“假”且具有最小时间片(time cell)的实体，并将该 time cell 作为下一最早事件发生时刻。注意：因为有可能有多个 B 活动在下一时刻发生，所以仿真控制程序必须记录在该时刻所有的不可用实体而形成一个 DueNow 列表。

(2) B 阶段：执行 DueNow 列表

一旦 DueNow 列表形成，仿真控制程序将顺序扫描列表中的每个实体，从中挑选出可执行的实体，对于每一个可执行实体进行如下操作：

① 将实体从 DueNow 列表中删除；

② 将该实体的“可用性”属性置成“真”；

③ 执行该实体“下一活动”属性所代表的活动。

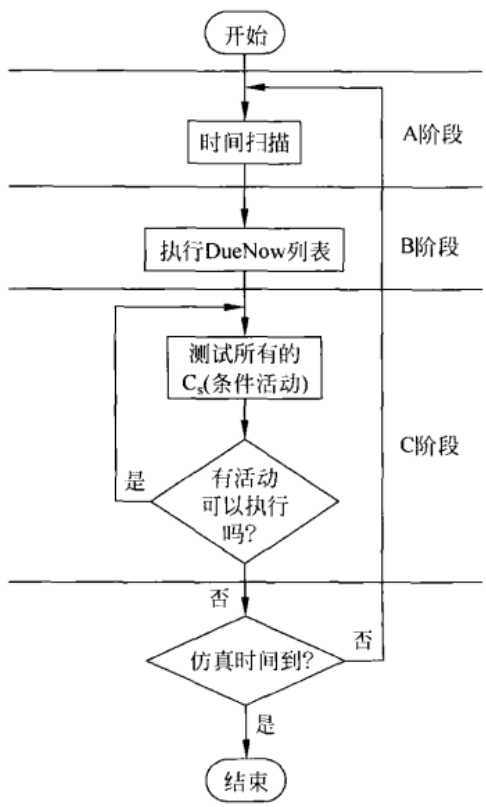


图 9.1 三阶段法的流程图

值得注意的是：执行相应的B活动将导致同一实体或其他实体归属于当前B活动或其他未来的B活动。

(3) C阶段：查询C事件表

逐一对其中的事件进行条件测试，看看其条件是否满足。如果条件满足，则执行相应的动作。在查询C事件表期间，保持当前仿真钟不变，直到所有的C事件都不满足启动条件。

例 9.1 订票员问题，航空公司售票处有职员一名，负责接待来访订票顾客和接听顾客咨询电话。为说明该系统的状态转移过程，我们画出其活动扫描图如图 9.2 所示。从活动的角度看，该系统有四种B活动和两种C活动，分别为：

B1：顾客到达　　B2：服务完成　　B3：来电话　　B4：交谈结束

C1：服务开始　　C2：开始交谈

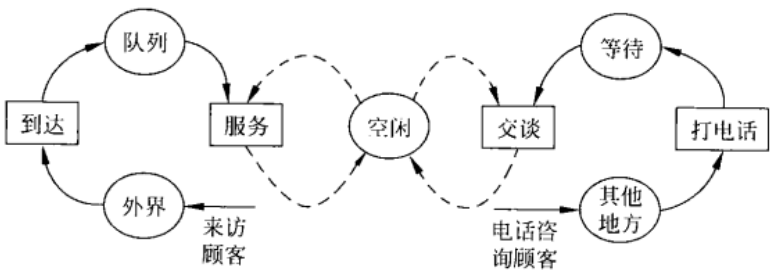


图 9.2 订票员问题的活动扫描图

在仿真过程中有三类对象：来访顾客、电话咨询顾客和售票处职员。将来访顾客和电话咨询顾客均视为临时实体而将职员视为永久实体，并增加另外一个新概念：到达机(arrival machine)用来模拟两类顾客的到达。在此，共有两类到达机：来访顾客到达机和电话咨询到达机。下面我们用三阶段法手工模拟一下该过程。

系统状态变量：

仿真钟：clock　　来访顾客总数：persIn

等待面谈顾客队列长度：queue　　电话询问总数：phoneIn

等待接电话的顾客队列：wait

系统初始状态：职员空闲(availability = *true*)；第一个来访顾客将于仿真钟时间=4时到达，第一个咨询电话将于仿真钟时间=6时到达(4，6这两个数都是由随机数发生器产生的)；仿真钟初始时刻为0；所有队列(等待面谈顾客和等待接电话的顾客队列)为空。

第一个A阶段

搜寻出那些“可用性”属性为“假”且具有最小时间片(time cell)的实体，并将该time cell作为下一最早事件发生时刻。由表 9.1 可知，这个实体应该是“来访顾客到达机”相应的time cell为4，因此A阶段结束时系统状态如表 9.2 所示。

表 9.1 clock=0：初始状态

状态变量：clock=0；queue=0；wait=0；persIn=0；phoneIn=0；DueNow列表为空。

实　体	time cell	availability	next activity
来访顾客到达机	4	*false*	来访顾客到达
电话咨询到达机	6	*false*	电话咨询到达
职员	0	*true*	

表 9.2 clock = 4：A 阶段结束

状态变量：clock=0；queue=0；wait=0；persIn=0；phoneIn=0；DueNow 列表为 1。

实 体	time cell	availability	next activity
来访顾客到达机	4	*false*	来访顾客到达
电话咨询到达机	6	*false*	电话咨询到达
职员	0	*true*	

第一个 B 阶段

执行在 DueNow 列表中的所有实体的 next activity 属性所代表的活动。因此在表 9.2 中可见，在 DueNow 列表中只有“来访顾客到达机”一个实体，它的 next activity 属性所代表的活动是“来访顾客到达”。执行该活动后，系统中的来访顾客数(persIn)变成 1，DueNow 列表中的实体数变成 0。接下来，来访顾客到达机的 time cell 属性被置成 9(假定下一顾客的到达间隔是 5 的话)。读者应当注意的是：在三阶段法中，我们应严格将 B 事件和 C 事件分开，尽管我们知道职员当前状态是空闲，但是服务并不在 B 阶段开始(因为开始服务是 C 时间，必须在 C 阶段开始)，因此该顾客被置于队列中。表 9.3 显示了第一个 B 阶段结束时的系统状态。

表 9.3 clock=4：B 阶段结束

状态变量：clock=4；queue=1；wait=0；persIn=1；phoneIn=0；DueNow 列表为空。

实 体	time cell	availability	next activity
来访顾客到达机	9	*false*	来访顾客到达
电话咨询到达机	6	*false*	电话咨询到达
职员	9	*true*	

第一个 C 阶段

查询 C 事件表，逐一对其中的事件进行条件测试，看看其条件是否满足。在这个例子中，有两类 C 事件，这两类事件的发生都需要两个条件：begin service 事件的发生条件是职员处于闲状态，并且有来访者排队等待服务；begin talk 事件的发生条件是职员处于闲状态，并且有顾客在等待接听电话。据此，begin service 时间可以满足启动条件。执行相应动作后，显示了第一个 C 阶段结束时的系统状态如表 9.4 所示。假定此次接待来访顾客的时间是 5，因此 time cell 属性将变成 9，availability 属性变成 *false*。状态变量 queue 变成 0。循环扫描 begin service 和 begin talk 的事件发生条件，均不满足，C 阶段结束。

表 9.4 clock=4，C 阶段结束

状态变量：clock=4；queue=0；wait=0；persIn=；phoneIn=0；DueNow 列表为空。

实 体	time cell	availability	next activity
来访顾客到达机	9	*false*	来访顾客到达
电话咨询到达机	6	*false*	电话咨询到达
职员	9	*false*	服务完成

读者可自己列出 clock=6、9 时，A，B，C 阶段的系统状态表，限于篇幅，在此不再赘述。

■

9.1.6 四种仿真策略的比较

前面我们对四种不同的仿真策略分别进行了讨论，本节我们对它们的特点进行比较。

(1) 系统描述

所有策略均提供主动成分及被动成分，每种成分均能接受其他成分的作用。在事件调度法中，只有主动成分 C_A 才能施加作用，而在活动扫描和进程交互两种策略中，主动成分 C_A 与被动成分 C_P 均可施加作用。

在事件调度法中，系统的动态特性表现为主动成分不断产生事件，而在活动扫描法中则表现为主动成分产生活动；在进程交互法中则是通过成分在其进程中一步一步地推进来描述；三阶段法则首先要明确系统中的实体，其次要定义系统中的B类活动和C类活动，并在这二者基础上维护系统的事件表。

(2) 建模要点

在事件调度法中，用户要对所定义的全部事件进行建模，条件的测试只能在事件处理子例程中进行。

活动扫描法设置了一个条件子例程专用于条件测试，还设置一个活动扫描模块，该模块对所有定义的活动进行建模。

进程交互则将一个进程分成若干步，每一步包括条件测试及执行活动两部分。

三阶段法要求严格按三阶段法的框图建模，DueNow 事件表的维护是仿真的关键。

(3) 仿真钟的推进

在事件调度法中，主动成分的下一事件发生时间保存在事件表中，定时模块不断地从事件表中取出具有最早发生时间的事件记录，将仿真钟推进到该事件发生时间，并转向该事件处理子例程执行。

活动扫描法除了设置系统仿真钟之外，每一个主动型成分还设有成分仿真钟。定时模块选择那些大于当前系统仿真钟的值且是所有成分仿真钟最小的那个成分仿真钟，然后将系统仿真钟推进到该时刻，并开始对活动扫描。

进程交互法采用将来事件表及当前事件表。当前事件表中的进程扫描完后，从将来事件表中取出具有最早发生时间的事件记录置于当前事件表中，仿真钟推进到该事件发生时间。一旦某个进程被执行，则要求尽可能多地走下去，但并不改变系统仿真钟；如果该进程并未完成，则将其断点记录下来，即将中断时间及事件类型放到将来事件表中，如果当前事件表中有一项或几项的发生时间小于当前系统仿真钟的值，则说明在以前的扫描中，发生该事件的条件未得到满足，本次应再次进行扫描。

三阶段法的仿真钟推进相对比较简明，搜寻出那些“可用性”属性为“假”且具有最小时间片(time cell)的实体，并将该 time cell 作为下一最早事件发生时刻。

(4) 执行控制

事件调度法：由定时模块按下一最早发生时间选择事件记录，并转向该事件处理子例程执行。如果下一最早发生事件不止一个，首先按事件的优先数来选择；对具有相同优先数的多个同时事件，解结规则由用户在建模时规定。

活动扫描法：按递减优先数的顺序对全部活动扫描，只有当 $D_{a_i}(S)=true$ 且 $t_{a_i} \leqslant TIME$

的活动才能被执行，对相同优先数的活动，解结规则由用户建模时加以规定。

进程交互法：按递减优先数的顺序对当前事件表的全部记录进行扫描，根据该事件在其进程中的指针进行条件判断，当 $D_{a_i}(S)=true$ 时则执行该进程（根据指针指示的位置），且只要条件满足，该进程一直执行下去，直到该进程结束；否则要记下断点，修改指针，并把该断点发生的将来时间登记在将来事件表中。当一个进程扫描完后，接着对下一优先数的成分进行扫描，直到扫描完毕。解结规则由用户在建模时定义。

三阶段法的框图清楚地表明了方法的控制过程，在此不再赘述。

概括地说，事件调度法建模灵活，可应用范围广泛，但一般要求用户用通用的高级语言编写事件处理子例程，建模工作量大。活动扫描法对于各成分相关性很强的系统来说模型执行效率高。但是，用户建模时，除了要对各成分的活动进行建模外，仿真执行程序结构比较复杂，其流程控制要十分小心。进程交互法是建模最为直观的策略，其模型表示接近实际系统，特别适用于活动可以预测，顺序比较确定的系统，但是其流程控制复杂，建模灵活性不如事件调度法。三阶段法兼有事件调度法和活动扫描法的优点，且控制流程清晰，易于程序实现，因而适用范围广阔。但实体的定义以及B类、C类活动的区分也需要建模者对系统有较为深入的理解和分析，这点不如进程交互法那样直观。

由此可以看到，这四种策略各有其优缺点，在离散事件系统仿真中均得到广泛的应用。有些仿真语言采用某一种策略，有的则允许用户在同一个仿真语言中用多种策略来建模，以适应不同用户的需要。显然，选择何种策略进行仿真建模依赖于被研究的系统的特点。一般说来，如果系统中各个成分相关性较少，宜采用事件调度法；相反则宜采用活动扫描法；如果系统成分的活动比较规则，则宜采用进程交互法或三阶段法。在具体编制离散事件系统仿真程序时，多数情况下不是单一地采用一种仿真策略，而是将这四种策略有机地结合起来使用。

9.2 离散事件系统仿真语言

9.2.1 离散事件系统仿真软件发展历程

离散事件系统仿真软件发展历程可分为6个阶段：

(1) 探索阶段

早期的仿真由FORTRAN或其他通用编程语言实现，缺乏专业仿真程序的支持。在最初阶段(1955年—1960年)，主要是探索统一的概念和开发可重用的例程，以促进仿真发展。

(2) 仿真语言出现阶段

仿真编程语言(SPL)出现于1961年—1965年。其中SIMSCRIPT和GASP是基于FORTRAN包、SIMULA和GPSS是ALGOL派生的。

GPSS(general purpose simulation system)由IBM的Geoffrey Gorden研发，并大约于1961年面世，是第一个进程交互仿真编程语言，加速了通信和计算机系统的仿真。基于方框图表示(类似于流程图)的GPSS适于各种排队模型。

Harry Markowitz(后来因其在最佳证券投资理论的工作获得诺贝尔奖)在1963年首次提出了SIMSCRIPT，RAND公司在美国空军的支持下开发出该语言。SIMSCRIPT最初深

受 FORTRAN 影响，但经过几个版本以后，开发人员将其从 FORTRAN 库中分离出来并建立了它自己的仿真编程语言，最初的版本是基于事件调度的。

1961 年，美国钢铁公司应用研究实验室的 Phillip J. Kiviat 开始开发 GASP(general activity simulation program，通用活动仿真程序)。最初是基于通用编程语言(ALGOL)，随后决定基于 FORTRAN。GASP 和 GPSS 一样，使用工程师们熟悉的流程图符号，它不是一个严格意义下的语言，而是 FORTRAN 例程集合，以辅助用 FORTRAN 进行仿真。

这一阶段还出现了许多其他的仿真编程语言。如 SIMULA(ALGOL 的扩展)开发于挪威并在整个欧洲广泛应用；另一个是控制和仿真语言(control and simulation language，CSL)，它采用活动扫描法。

(3) 仿真语言形成阶段

1966 年—1970 年间，主要的仿真编程语言更加成熟，获得了更加广泛的应用。

硬件的飞速发展和用户需求迫使一些语言，尤其是 GPSS，经历了重要修订。GPSS/360 以及其早期版本的扩展，是应 IBM360 计算机而出现的。

SIMSCRIPT Ⅱ代表了仿真编程语言的重要发展，它有格式自由的类英文语言以及“包容性”好的编译器。

CSL 派生出 ECSL，ECSL 发展并流行于英国。

在欧洲，SIMULA 增加了类和继承的概念，并成为现代面向对象编程语言的先驱。

(4) 仿真语言发展阶段

1971 年—1978 年间，GPSS 除了主要来自 IBM 以外，Norden Systems 的 Julian Reitman 领导了 GPSS/NORDEN 的开发，它提供了一个交互式、可视化在线环境。1977 年，Wolverine Software 的 James O. Henriksen 为 IBM 主机开发出 GPSS/H。据称它可以比标准 GPSS 快 5 到 30 倍的速度编译。

1974 年，Purdue 大学的 Alan Pritsker 对 GASP 做出了重要改动，推出了 GASP Ⅳ。除了时间事件外，它结合了状态事件，从而除了事件调度世界观外，增加了对活动扫描世界观的支持。

这一阶段人们还努力简化建模过程。通过 SIMULA 人们尝试从高级用户的视角开发一个系统定义，以便可以自动翻译为可执行模型。

(5) 仿真语言巩固和改进阶段

1979 年—1986 年间，出现了专为桌面计算机和微机所写的 SPL，或可用于桌面计算机及微机的 SPL。同时，SPL 扩展到许多计算机和微处理器。

SLAM Ⅱ和 SIMAN 由 GASP 派生，Pritsker 和 Associates 公司推出了面向建模的仿真语言(SLAM)，对提供多种建模观点和组合建模的软件进行了探索。

SIMAN(SIMulation ANalysis)既具有像 GASPⅣ这样的 SPL 的通用建模能力，又具有某些方面类似 SLAM 和 GPSS 的块图组件。C. Dennis Pegden 用两年时间开发了 SIMAN，随后他成立了系统建模公司将 SIMAN 推向市场。SIMAN 是第一个可在 IBM PC 上执行，并专为在 MS-DOS 系统下运行而设计的主流仿真语言。类似于 GASP，SIMAN 可以用事件调度法，采用 FORTRAN 编程，并提供了一个 FORTRAN 程序集。

(6) 仿真集成环境阶段

自 1987 年以来，因为 SPL 在个人计算机上的迅速发展和具有图形化用户界面、动画以

及其他可视化工具的仿真环境的出现，一些软件包试图通过使用进程流或块图，以及使用填空式视窗系统而无需学习编程语法，来简化建模过程。

目前离散事件系统仿真软件分为三类：第一类，通用的编程语言，如C、C++和Java等；第二类，仿真编程语言，如GPSS、SIMAN和SLAM等；第三类，仿真环境，如图形化用户界面以及支持仿真研究的所有（至少是大多数）方面的环境。

第一类仿真软件，如Java，是一种通用编程语言，不是为仿真应用而专门设计的。目前很少有人编写离散事件仿真模型只采用通用编程语言，然而在一些应用领域，一些人仍然在使用基于Java或其他通用语言的包。

第二类仿真软件，如GPSS/H，是一种高度结构化的进程交互仿真语言。GPSS曾被设计用于排队系统相对简单的仿真，但它已经用于仿真高度复杂的系统。GPSS首先为IBM所引入，目前，GPSS有多种实现方案，GPSS/H是其中应用最为广泛的一种。

第三类仿真软件，如Arena，是一种集成交互的仿真环境。Arena采用了基于对象的设计，完全图形化的建模环境。Arena可以用于建立离散/连续系统的复合模型，如利用其内置连续建模能力来建立药品和化学品生产的模型。

9.2.2 典型仿真语言简介

1. GPSS

GPSS语言也许是离散事件系统仿真语言中推出最早，使用最为广泛的一种。它是一种基于进程交互方法和面向排队系统的高度结构化、专业化的仿真编程语言。

GPSS最大的特点是仿真模型可采用一组标准的程序块（block）来表示其逻辑结构。block图提供了一种方便的方法来描述被仿真的系统，称为事务的实体可被视为流经block图，block用来表示事件、延迟和其他影响事务流动的动作。GPSS的标准块超过40种。系统的临时实体按程序块图中规定的方向由某一块流向另一块。一个程序块对应一条GPSS语句，它相当于一个子程序，因而具有较强的功能，使得GPSS具有较强的建模能力。当用户用图形描述出一个系统的仿真模型后，根据其流程用GPSS语句写出其相应的GPSS源程序就可以了。因此，对用户来说，块图转换成方块语句，加上控制语句，其结果就是一个GPSS模型，只需掌握GPSS语句就能很方便地对系统建立仿真模型。GPSS可用于对任何事务（实体、顾客、交通单元）流经一个系统（例如，由队列处理稀有资源的队列网络）的场合进行建模。

GPSS的第一个版本是IBM于1961年发布的。它是第一个进程交互仿真语言，因而得到流行。自1961年以来，其许多部分得以重新实现和改进，因而有多种版本。不同的GPSS版本所包括的程序块个数不尽相同。表9.5列出了几种典型GPSS版本所包括的程序块数目。

表9.5 不同GPSS版本的程序块数

GPSS版本	程序块数目
GPSS V	48
GPSS/X	56
GPSS/66	53
GPSS/1100	51
GPSS/PC	38

GPSS/H是当今使用最广泛的版本，由美国Wolverine软件公司开发的产品（Banks，Carson and Sy，1995；Henriksen，1999），它是一个适应性强、功能强大

的仿真工具。与原来 IBM 实现的版本不同，GPSS/H 包括内置文件和屏幕 I/O，使用算术表达式作为块操作数，拥有交互式调试器、更快速的执行能力、扩展控制语句、普通变量和数组、浮点计时器，内置了数学函数和随机变量发生器。

GPSS/H 的动画生成器是 proof animation™，它是 Wolverine 软件公司(Henriksen，1999)的另一个产品。proof animation 提供了 2D 动画，通常画面基于原物成比例形成。它能用后处理模式(在仿真完成运行后)或并发模式运行。在后处理模式下，动画由两个文件驱动：用于静态背景的分层文件和包含使对象移动或产生其他动态事件的命令的跟踪文件。它可以与任何可写 ASCII 跟踪文件的仿真软件包一起运行。另一种模式是并发模式，它运行仿真时，可同时将跟踪文件命令作为消息发送出去，而且通过其 DLL(动态链接库)版本能直接接受控制。

用 GPSS 进行仿真建模时，要进行如下三个步骤：

① 将被仿真的系统抽象成为类似于流程图的模型描述；

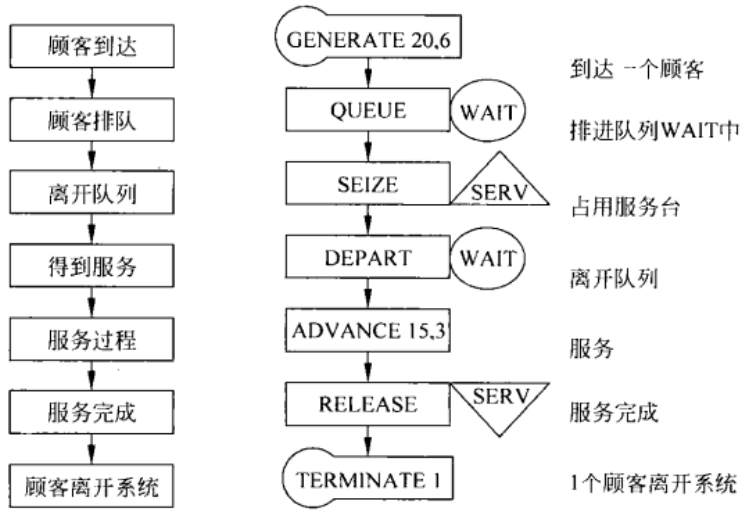
② 用 GPSS 程序块实现流程图中的功能模块；

③ 用 GPSS 程序块对应的语句编写出仿真程序，形成一个可执行的代码。

下面通过一个简单的例子来说明如何用 GPSS 对离散事件系统进行仿真。

例 9.2 单台排队系统，顾客到达事件间隔为均匀分布的随机变量，其均值为 20min，偏差为±6min；为每个顾客服务的时间也是均匀分布的随机变量，其均值为 15min，偏差为±3min；服务规则为先来先服务(FIFO)。

图 9.3(a)和(b)分别给出了该系统的流程图描述及 GPSS 程序模型图描述，对这类简单系统，它们之间几乎是一一对应的。GPSS 程序块图旁边是注释。



(a) 流程图　　(b) GPSS模型图

图 9.3 单排队系统

然后就可以写出该系统的 GPSS 程序如下：

```
* The Model of Single-Server Queue System
        Simulate
```

```
* Block Definition Cards
        Generate    20,6
        Queue       WAIT
        Seize       SERV
        Depart      WAIT
        Advance     15,3
        Release     SERV
        Terminate   1
* Control Cards
        Start       100
        End
```

该程序中，前面冠以“*”的为注释语句，仅起说明作用，以 Simulate 语句开始表示仿真执行开始。程序分为两部分，第一部分为模型定义，第二部分为实验控制。本程序是以 Start 100 语句实现控制的，它表示仿真运行服务 100 个顾客结束，与 Terminate 1 语句相配合，每执行一次 Terminate 1，则计数器从 100 开始逐次减 1，直到 0 为止则结束仿真。其他几乎可以不加说明就可以基本理解。程序最后为 End 语句，表示 GPSS 程序到此结束。

■

GPSS 语言提供一个处理器以翻译 GPSS 源程序并加以执行，最终向用户输出仿真结果的标准报告。遗憾的是，GPSS 的仿真钟只能取整数值，因此必须根据仿真精度的要求选择仿真钟的基本时间单位。这就意味着只能在基本时间单位整数倍的时间点上发生系统状态的变化。例如一个 M/M/1 系统，若选择时间单位为 min，那么顾客的到达时间间隔或服务时间长度必须是 min 的整数倍。这样导致了在系统中状态变化同时发生的概率增加，排序处理就十分重要。GPSS 对每一类实体都专设一个属性——优先组（范围为 0～127），值越大，则优先级别越高。这对于那些难以确定优先级的系统来说，无疑会增加建模的难度。有关 GPSS 的详细内容请读者参见有关书籍。

2. SLAM/SIMAN 仿真语言

SLAM 语言是另一种应用十分广泛的仿真语言，可用于离散事件系统、连续系统以及离散-连续混合系统的仿真。它将事件调度及进程交互两种策略结合起来形成了统一的建模框架。连续系统既可采用微分方程，也可采用差分方程来描述。离散部分若采用面向进程的观点建模则采用网络结构，网络模型中的实体可以触发离散事件的发生及连续状态变量的值发生变化；若采用面向事件的观点建模，则事件可以改变网络模型中的实体流动并可引起连续状态变量发生变化；而达到阈值的连续状态变量能激发网络模型中的实体或触发事件。SLAM 共提供了 22 个网络语句，与 GPSS 相似，每个语句对应有一个称为“节点”的图形描述或用带箭头的分支符号表示。如果用户还需要用事件调度的观点来描述系统，SLAM 提供了 EVENT(I)子程序名，用户可用 FORTRAN 编写各事件处理子程序；此时还需用户编写主程序（基于 FORTRAN）并在主程序中执行 CALL SLAM，以便进入其网络模型。

SLAM 语言提供运行程序负责对网络语句进行翻译，执行并生成标准的输出报告。如果用户模型中有用 FORTRAN 编写的 EVENT(I)，则必须事件将这些子程序连同主程序进行 FORTRAN 编译，然后与 SLAM 库子程序进行连接后才能运行。

SLAM 的网络模型与 GPSS 的方块图基本相似，但是它的优点是仿真钟可取实型值，同时允许在同一个模型用两种观点来建模。这比 GPSS 具有更大的灵敏性。

SLAM 源程序可分为两大部分：一部分是系统模型描述；另一部分是实验控制语句。这种将模型语句与实验控制语句组织在一个仿真程序中的做法是早期仿真语言共同的特点。后来，建模与实验分离的思想日益得到重视，其最大的优点是如果模型不变，而仅仅是实验条件变化时，用户无需修改仿真源程序，实现了一个模型多种实验的要求，极大地提高了仿真的效能。根据这一思想，SLAM 的作者之一 C. Dennis Pegden 对 SLAM 进行了较为彻底的改造与扩充，研制了 SIMAN 语言。

SIMAN 语言将仿真全过程分成了四个阶段，可以用图 9.4 来表示。

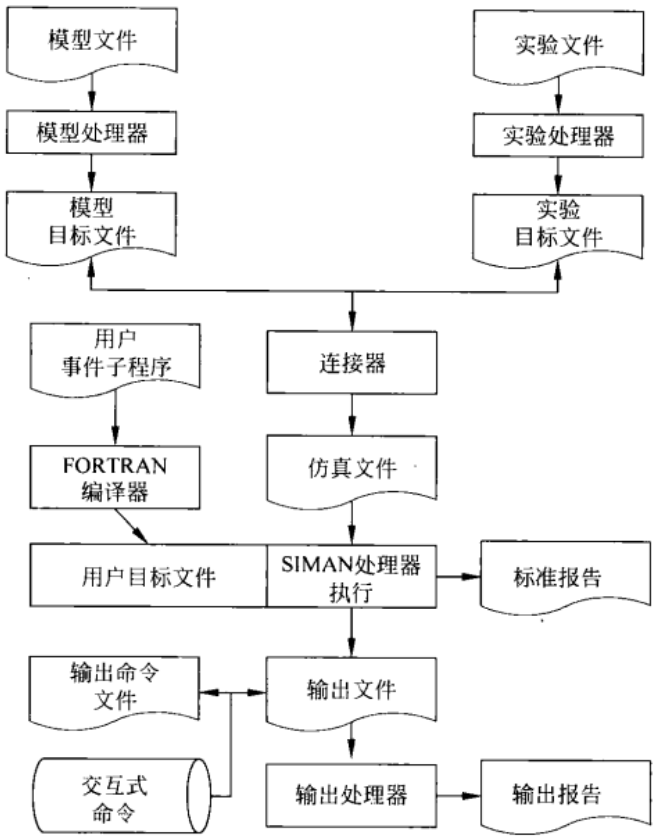


图 9.4 SIMAN 语言仿真结构图

1）进程模型处理

SIMAN 的进程模型与 SLAM 的网络模型十分接近（增加了某些功能）。它将该模型描述作为一个单独的文件——进程模型文件，并有一个模型处理器 MODEL 对该文件进行翻译，得到一个相应的模型目标文件。

2）实验框架处理

SIMAN 将用户的实验要求用实验框架文件来描述，而且与模型完全分离开来。这样，一个模型文件可用于若干个不同的实验框架文件。实验框架处理器 EXPMT 负责对实验框架文件进行翻译，产生一个相应的实验框架目标文件。

3）仿真运行

SIMAN 语言设计了一个连接器 LINKER 将上述两个目标文件连接起来形成一个仿真文件，该文件可提交给 SIMAN 处理器执行，并自动产生标准输出报告。如果用户编写了事件处理子程序，则必须事先进行 FORTRAN 编译，生成用户目标文件，SIMAN 会将它们与仿真文件连接起来一起执行。

4）仿真输出

SIMAN 语言除了可提供标准输出报告以外，还根据实验框架中规定的内容产生用户输出文件，用户可交互式地或批处理方式对该输出文件进行各类数据处理。

SIMAN 的另一个重要应用是提供了专用于制造系统的语句，因而在制造系统仿真中得到越来越广泛的应用。

3. SIMSCRIPT II. 5

仿真过程中仿真建模是最费时费力的。前面介绍的 GPSS、SLAM、SIMAN 语言各自提供了自身的语句以减轻用户建模的工作量。但是，任何一种仿真语言它所能提供的语句总是有限的，然而被仿真的实际系统却是千差万别的。要将这些系统用这些语言的语句来描述仍然要付出巨大的努力，有时只好借助于高级语言来描述那些难以用仿真语言规定的语句描述的系统。近些年来，仿真语言的另一个发展就是非语言(non-language)建模技术，即不是采用程序设计语言或仿真专用语言的语句来建模。

非语言建模技术有各种各样的形式，如图形建模、数字化仪输入、正文建模等。SIMSCRIPT II. 5 就是一种非语言建模的仿真语言。它采用近似英语的自然式句法定义模型，从而程序易于表达且便于阅读，并可以文件的形式加以保存。

SIMSCRIPT 最初版本采用事件调度法建模，在以后的版本中(如 II. 5)加进了“进程”与“资源”等概念，从而也允许用进程交互法建模。

SIMSCRIPT 的仿真模型由三部分组成：前言、主程序、事件(或/与)进程子程序，其描述方式近乎自然语言，具有非程序语言建模能力。

4. Arena

Arena Basic，Standard 和 Professional Editions 是系统建模公司的三个版本，Arena 可用于仿真离散和连续系统，OptQuest for Arena 是 Arena 产品系列的一个优化软件包。

Arena Basic Edition 是对经营过程和其他系统建模，满足高层分析需求，用一个递阶流图表示进程，用电子数据表保存系统信息，它内置基于活动的成本计算，并和流程图软件 Visio 紧密集成。

Arena Standard Edition 是为离散和连续系统更细化的模型而设计的。1993 年发布的第一个版本中，Arena 采用了基于对象的设计、完全图形化的模型开发。从图形化对象调用模块来建立仿真模型，以定义系统逻辑以及物理组件，诸如机床、操作员及职员等。模块用图标加上在对话框输入的相关数据来表示，这些图标相互连接起来表示实体流。模块则被组织成集合，称为模板。Arena 模板是为建立各种类型的应用模型提供通用特征的核心集合。除资源、队列、进程逻辑和系统数据等标准特征外，Arena 模板还包括了用于制造和物料储运系统的专用模块。

Arena 的核心是 SIMAN 仿真语言，是动画显示仿真模型，其核心模型结构带有显示队列、资源状态和实体流的标准图形。

为了表示已有数据，比如进程和到达间隔时间，Arena 的 input analyzer 实现了选择合适的分布及其参数的过程自动化。

output analyzer 和 process analyzer 自动地比较各种不同设计方案。

5. AutoMod

AutoMod Product Suite 是 Brooks Automation 推出的，包括仿真包 AutoMod、用于实验和分析的 AutoStat 以及用于制作内置 3D 动画的 AVI 电影的 AutoView。AutoMod 更关注制造和物料储运系统。

AutoMod 有内置模板，可用于大多数常见物料储运系统，包括运输小车系统、传送带、自动存储和取回系统、桥式起重机、有动力/无动力传送带。通过其 tanks 和 pipes 块，AutoMod 支持液体和散装材料流的连续建模。

AutoMod 还包括了一个完整的仿真编程语言，它的 3D 动画可实时地从任何角度或视点进行观察。用户可以自由缩放、摇镜头或旋转 3D 空间。

6. Flexsim

Flexsim 仿真软件是美国犹他州 Orem 的 Flexsim 软件产品公司开发的。Flexsim 是用 C++开发的离散事件、面向对象的仿真器，采用了 OpenGL 技术。动画可以以树形视图、2D、3D 以及虚拟现实来显示。所有的视图都可以在运行阶段的模型开发期间并行地进行显示。它集成了 Microsoft 的 Visual C++ IDE 以及带有图形化 3D 点击-拖动式仿真环境的编译器。

Flexsim 软件用于通过仿真、实验和优化来改进动态系统的生产效率、减少操作成本。工程师和管理者使用 Flexsim 可以评估制造厂的生产能力、平衡包装和生产线、管理瓶颈、解决在制问题、计算资本消耗、安排设备维护计划、建立合适的仓储水平、改进订单接收系统以及优化生产率。

Flexsim 允许最终用户引入模型新条件并进行仿真，从而分析它们的影响和结果以寻找改进所研究系统的方法。使用 Flexsim，效率-生产量的提高和成本的降低——都可得以确认、测试并在应用于真实系统之前被验证。每个仿真的结果都可以通过 3D 动画、统计报告和图表进行分析。

7. ProModel

ProModel 是 PROMODEL 公司开发的，是建立制造系统模型的仿真和动画工具。该公司还为医疗系统提供了 MedModel 以及为服务系统提供了 ServiceModel。ProModel 提供了带有 3D 视角选项的 2D 动画，ProModel 的动画在模型开发时自动生成。

ProModel 具有面向制造业的建模元素和基于规则的决策逻辑。一些系统可以通过从高度参数化的模型元素集合 ProModel 中选择建立。另外，它的仿真编程语言提供实现特殊环境的建模。

在 ProModel 中，建模元素是零件(实体)、位置、资源、路径网络、路由和加工逻辑以及

到达。零件到达并沿着路由和加工逻辑从一个位置到另一个位置。用资源表示人员、工具或车辆，车辆在位置之间运送零件、在一个位置执行零件加工、在一个位置进行维护或补充耗尽的资源。资源以给定的速度、加速度在路径网络移动并获取或设置移动时间。路由和加工元素允许用户用 ProModel 仿真编程语言定义过程逻辑。

ProModel 包括自动生成与某一过程相关的费用数据的逻辑，可以为位置使用、资源和实体添加费用。ProModel 完成时有一个输出视图，允许直接用数据表示，以及用图形及图表表示，例如状态图。ProModel 运行时界面允许用户为实验定义多个场景。

8. QUEST

QUEST(queueing event simulation tool)是 Delmia 公司的软件，是一个面向制造业的仿真软件包。QUEST 将一个基于对象的、真实 3D 仿真环境、图形化用户界面和物料流块相结合以对工人、传送带、自动导引小车、运动设备、起重机、动力和无动力传送带以及自动存储和取出系统建模。QUEST 模型把 2D 和 3D CAD 几何形状结合起来以建立虚拟工厂环境。

Delmia 还提供许多工作单元仿真器，包括：用于机器人仿真和编程的 IGRIP 以及用于人体工效学分析的 ERGO。在 IGRIP 和 ERGO 中进行仿真的机器人和自然人工作单元都可以可视化地、数值化地加载到 QUEST 模型中。

Delmia 通过 PROCESS ENGINEER™(Delmia 的工艺规划环境)与 QUEST 以及其他制造技术提供进一步的集成。在该软件背后的 manufacturing hub 基础结构由一个面向对象的数据库构成，以存储 product 对象、process 对象和 resource 对象，这些对象都经过配置管理且实现了有效控制。

对于唯一性问题，可以使用 Delmia 的 QUEST 仿真控制语言(SCL)。这种结构化仿真编程语言提供了访问所有系统变量的分布式处理方案。SCL 允许专家级用户定义客户的行为并获得对仿真的全面控制。

Delmia QUEST 的开放式结构允许高级用户进行批处理仿真运行以通过 batch control language(批控制语言，BCL)自动收集数据并制作成表格。

输出可以采取数值化方式(通过统计报告机制)或可视化方式(通过所得到的虚拟类工厂动画)。统计输出结果既可以通过图形化用户界面从内部得到，又可以通过 HTML 从外部得到，还可以通过使用 XML 或 QUEST 自己的 BCL 定制。

9. SIMUL8

1995 年，SIMUL8 是 SIMUL8 公司推出的，是利用一系列代表系统中的资源和队列的图标与箭头，通过计算机鼠标绘制工作的流程，从而建立模型。对图标的所有属性都提供默认值，因此在建模过程的早期就可以观察到动画。SIMUL8 主要关注的是人们处理交易的服务业。

和其他软件包一样，SIMUL8 具有“模板”和“组件”的概念。模板或者说预建的仿真，着重于特定的重复发生的决策类型，这种类型以迅速的参数化以适应一个特定公司的要求。组件则是用户定义的图标，该图标可以重用并在一个公司的仿真中进行共享。这样就减少了建立仿真的时间，将一个公司之内的情境处理都标准化，并免去了仿真研究中许多数据收集阶段。

SIMUL8 公司对交易的处理方法和大多数其他软件包不同，它们宣称其目标是在交易

中广泛的推广仿真，而不是局限于专业的、高度训练的仿真专业人员范围内。这意味着它们拥有截然不同的价格和支持策略，但同样也意味着其软件不得不包含一些特征。

SIMUL8 将仿真模型和数据存为 XML 格式以使其易于和其他应用程序之间的相互转换。它提供了一些非仿真特性，以使它可能为建模者用电子数据表、对话框或向导表建立客户化的用户界面。

本产品有标准版和专业版两个等级的版本。两个等级版本都提供同样的仿真特性，但专业版增加了仿真的 3D、“虚拟现实”视图，以及数据库连接以实现数据库协同，还具有一些只对专业仿真建模人员来说可能有用的特点。

10. WITNESS

WITNESS 由 Lanner 集团提供并有分别用于制造业和服务业的独立版本，包括用于离散零件制造业的许多元件，也有用于连续处理的元件，如经过处理器、储水池和管道的液体流。

WITNESS 模型基于模板元件，这些可以定制并组合而成为模块元件和模板以便重用。标准机床元件可能是单台的、成批的、加工型的、装配型的、多站的或多周期的。其他离散模型元件还包括多种类型的传送带、轨道、车辆、工人和搬运装置等。在 WITNESS 用户界面中每个元件的行为特性都用一个带标签的详细表格描述。

模型用 2D 分层动画显示，多窗口和多显示层；可选择采用过程流显示或者元件-路由覆盖图。可以在模型运行的任何时刻修改模型并在任何运行点保存以便以后重载使用。

可选择的 WITNESS 模块还包括 WITNESS VR，工作模型的一个集成的虚拟现实 3D 视图，该视图可以完全用鼠标控制摄像角度和位置。还有，为后期处理 VR，在带有多屏幕投影和各种不同的头戴式耳机技术方面也有多种选择项。

WITNESS 为嵌入式仿真提供了对象-模型和 ActiveX 控制，并包括与 Microsoft Excel、MINITAB 以及任何其他 OLEDB 数据源的直接数据连接。另外还有 XML 数据格式保存提供另外的连接功能。

9.3 小结

在一个较为复杂的离散事件系统中，一般都存在诸多的实体，这些实体之间相互联系，相互影响，然而其活动的发生即统一在同一时间基上，采用何种方法推进仿真钟，建立起各类实体之间逻辑联系，这是离散事件系统仿真建模方法学重要内容之一，有时称之为仿真算法或仿真策略。

如同连续系统仿真一样，即使是同一系统，不同算法下的仿真模型的形式是不相同的，仿真策略决定仿真模型的结构。本章介绍了目前比较成熟的四种仿真策略，即事件调度法、活动扫描法、进程交互法和三阶段法。在建立仿真模型时，一般并非只拘泥于采用某一种策略，同一个模型中可采用几种策略。但为了讨论方便，以便读者对每一种策略的实质及其机理有比较深刻的认识，我们分别对每种策略进行分析。

由于离散事件系统的复杂性，仿真作为这类系统的分析工具使用日益广泛。用于开发仿真模型的软件可以划分为三类：第一类，通用的编程语言，如 C、C++ 和 Java 等；第二类，仿真编程语言，如 GPSS、SLAM/SIMAN 和 SIMSCRIPT 等；第三类，仿真环境，这一类包

括 Arena、AutoMod、ProModel、Flexsim、WITNESS 和 QUEST 等多种产品，它们都具有图形化用户界面，以及支持仿真研究的各个方面的环境。许多仿真环境包括一种仿真编程语言，但有些则采用类似于流程图的图形化方法。本章介绍了离散事件仿真语言的发展历史，对国内外使用得比较有代表性的几种仿真语言的特点进行综述性的说明，重点介绍了 GPSS 语言，对某种仿真语言特别感兴趣的读者可进一步参考有关文献或手册。

习题

1. 试分别用事件调度法、活动扫描法、进程交互法对 9.1.5 节中讨论的订票员问题进行仿真描述(即用相应的算法，描述仿真过程)。

2. 在一个大城区，救护车以每 15±10min 一辆的速率派发。15%的求救呼叫是假报警，这些报警需要 12±2min 完成。所有其余的呼叫可以分为两种情况。一种是严重的，在非假报警中占 15%，并且需要 25±5min 完成。剩下的呼叫需要 20±10min 完成。假设现有足够数目的救护车，任何时间都有充足车辆应答。仿真这个系统直到有 500 个呼叫完成。

3. 一个单座男女皆宜的理发馆，到达速率为每 20±15min 一位。到达顾客中有 50%要全剪，30%要做发型，20%只要单剪。全剪一次花 15±5min，做发型一次花 25±10min，单剪一次花 10±3min。对 400 个顾客通过该理发馆的情况进行仿真。将给定的服务种类的比例与仿真结果进行比较，结果可信吗？

4. 一个机场有两个大厅。大厅 1 的乘客以每位 15±2s 的速率到达，大厅 2 以每位 10±5s 的速率到达。通过大厅 1 需要费时 30±5s，通过大厅 2 需要费时 35±10s。两个大厅都通向与行李领取处相邻的主休息大厅。从主休息大厅到行李领取处需要费时 10±3s。只有 60%的乘客将去行李领取处。对 500 位顾客通过机场系统进行仿真。多少乘客到过行李领取处？本题中，去行李领取处的乘客期望数是 0.60×(500)=300。仿真估计值和期望值有多接近？差异原因是什么？

5. 假设零件由传送带运送，传送速度为每天的头 3 小时 15±5s 一个。机器 1 在第 1 小时进行加工，机器 2 在第 2 小时进行加工，机器 3 在第 3 小时进行加工。每天可以生产多少个零件？需要为每个机器提供多大的存储区以存放等待的零件？在任何特定的时间，零件是否发生堆积现象？为什么？

6. Lower Altoona 大学只有一台计算机为学生使用。学生以 15±10min 一位的速度到达计算机，并使用计算机 12±6min。若计算机忙，60%的学生将在 10min 后回来重新使用计算机。若计算机仍忙，这部分(60%)学生中 50%将在 15min 后再次返回。在 500 名学生实际完成使用计算机的情况下，有多少学生最终未能使用计算机？一天 24 小时都有请求和服务发生。

7. 一座仓库可以存放 1 000m^3 箱子。这些箱子有三种型号：小型($1m^3$)、中型($2m^3$)和大型($3m^3$)。箱子以下列速率到来：小型，每 10±10min 一个；中型，每 15min 一个；大型，每 8±8min 一个。如果不移除任何箱子，填满一个空仓库需要多长时间？

8. 零件以均值为 60s 的指数分布的时间间隔的随机方式到达机器。所有的零件都需要 5s 时间的准备和与机床对准。一共有 3 种不同类型的零件，它们的比例见下表。加工每种零件的时间呈正态分布，均值和标准差也如下表。

零件类型	比例	均值/s	标准差/s
1	50	48	8
2	30	55	9
3	20	85	12

求完成所有类型零件加工的总时间的分布。完成加工的时间大于 60s 的比例有多大？零件等待的平均时间有多长？仿真一个 8 小时工作日。

9. 在一个百货公司的购物时间呈以下的分布：

购物时间/min	购物人数	购物时间/min	购物人数
0～10	90	30～40	145
10～20	120	40～50	88
20～30	270	50～60	28

购物后，顾客可以选择 6 个收款台之一来付款。付款时间服从均值为 5.1min，标准差为 0.7min 的正态分布。到达时间间隔服从均值为 1min 的指数分布。对每个收款台收集统计数据(包括等待付款的时间)。用表格表示完成购物的时间分布以及完成购物与结账的时间分布。顾客在商场花费 45min 以上的比例是多少？仿真一个 16 个小时工作日。

参考文献

[1] Pritsker A A B. Introduction to Simulation and SLAM Ⅱ. John Wiley & Sons, Inc. 1984

[2] Hooper J W. Strategy-related Characteristics of Discrete event Languages and Models. SIMULATION,1986, 4(46): 153～159

[3] O'Keefe R M. The three-phase approach: A comment on "Strategy-related characteristics of discrete-event languages and models". SIMULATION,1986,5(47): 208～211

[4] Zeigler B P. Theory of Modeling and Simulation. John Wiley & Sons,Inc. ,1976

[5] 宋承令. 离散事件仿真策略与结构模型. 系统仿真,1987,1

[6] Law A M,Kelton W D. Simulation Modeling and Analysis. McGraw-Hill,1982

[7] Pritsker A A B. The GASP IV Simulation Language. Wiley,New York,1974

[8] CACI,Inc.. SIMSCRIPT II.5 Reference Handbook. LOS Angeles,1976

[9] Gordon G. System Simulation. Prentice-Hall,Englewood Cliffs,N. J. ,1978

[10] Pegden C D. Introduction to SIMAN. Systems Modeling Corporation,1982, 12

[11] Peterson J L. Petri Net Theory and Modeling of System. Prentice-Hall,1981

[12] Banks J,Carson J S 著. 肖田元,范文慧译. 离散事件系统仿真. 北京：机械工业出版社,2007

[13] www. arenasimulation. com

[14] www. delmia. com 和 www. 3ds. com

[15] www. imaginethatinc. com

[16] www. flexsim. com

[17] www. maad. com

[18] www. promodel. com

[19] www. simuls. com

[20] www. witness-for-simulation. com

第 10 章
离散事件系统仿真输出分析

输出分析是对仿真所产生的数据进行检查，它的目的是预测一个系统的性能或比较两个或多个备选系统设计的性能。本章讨论的内容是这样安排的，在 10.1 节，首先讨论仿真输出分析方法的分类，然后在 10.2 节及 10.3 节中分别讨论单个系统的终止型仿真与稳态型仿真的分析方法；在 10.4 节，讨论多个系统仿真结果的比较技术；在 10.5 节，通过一个实际例子综合运用本章中介绍的几种方法，以加深读者对仿真结果分析的理解；在 10.6 节介绍提高仿真分析效能和精度的方差减少技术。

10.1 仿真输出分析方法的分类

通过前面几章的介绍，我们对离散事件系统仿真有了比较详细的了解。在这些章节中，我们还多次提过，由于离散事件系统固有的随机性，每一次仿真运行仅仅是系统模型输出的一次抽样，所得到的结果与系统的“真正”解可能有很大的方差。

例 10.1 考虑 M/M/1 系统，顾客到达时间间隔是均值为 5min 的指数随机变量，为每个顾客服务的时间是均值为 4min 的指数随机变量。采用仿真方法，分别对长度为 $n=1\,000,2\,000,3\,000,4\,000,5\,000$ 个顾客服务完毕进行仿真，其平均队长 $Q(n)$ 及平均等待时间 $d(n)$ 如表 10.1 所示。

表 10.1 M/M/1 系统仿真结果

仿真长度(n)	理论值	1 000	2 000	3 000	4 000	5 000
$d(n)$	16.0	19.723	17.856	15.563	16.826	16.982
$Q(n)$	3.2	3.916	3.620	3.181	3.326	3.425

由表 10.1 可以看到，只有当 $n=3\,000$ 时最接近稳态理论值，其他各次运行的结果相对来说离理论值较远。可见仅从某一次仿真运行的结果来推断系统的性能并不一定保证是正确的。■

这就提出了这样一个问题，在离散事件系统仿真研究时，如何恰当选择仿真运行长度才能使仿真结果接近被仿真系统的实际性能呢？或者说，既然一次运行仅仅是随机变量的一次抽样值，那么在仿真研究时，如何控制仿真运行次数才能使估计值接近实际值呢？这就是本章要讨论的仿真结果分析问题。

从仿真结果分析的观点来看，仿真运行方式可分为两大类。

第 1 类称为终止型仿真，这种仿真的运行长度是事先确定的，例如在第 7 章中讨论的单人理发馆系统，其仿真时间长度为 8 小时。由于仿真运行时间长度有限，系统的性能与运行长度有关，系统的初始状态对系统性能的影响是不能忽略的。显然，理发馆开门时队长为 0 与有若干个顾客排队，对于在 8 小时之内服务的顾客数是有影响的。

为了消除由于初始状态对系统性能估计造成的影响，需要多次独立运行仿真模型，从统计学的观点来看，理论上要独立运行无穷多次，因为每次运行仅仅是系统性能估计值的一个样本。在实际中如何确定运行次数以便得到较好的性能估计，这是终止型仿真结果分析需要讨论的问题。

例 10.2 考虑 M/M/1 系统(选择 M/M/1 系统是因为可解析计算系统的有关性能)，通过仿真估计顾客平均排队等待时间。

记 D_i 是第 i 个顾客在队列中的等待时间，仿真运行长度为服务 m 个顾客结束，初始队长为 0，服务台初始状态为闲，则顾客平均排队等待时间的估计值为

$$d(m \mid l(0)=0)=E\left(\frac{\sum_{i=1}^{m} D_i}{m}\middle| l(0)=0\right)=\lim_{K\to\infty}\left(\frac{\sum_{j=1}^{K}\overline{D}_j}{K}\middle| l(0)=0\right) \tag{10.1}$$

其中，$\overline{D}_j$ 是第 j 次运行时顾客排队等待时间 D_{ji} 的平均值，即

$$\overline{D}_j=\sum_{i=1}^{m} D_{ji}/m \tag{10.2}$$

实际上，K 不可能无穷大，需要研究如何由有限的 K 次运行得到 d 的好的估计值。

■

第 2 类仿真叫稳态型仿真。这类仿真研究仅运行一次，但运行长度却是足够长，仿真的目的是估计系统的稳态性能。显然，由于仿真长度没有限制，系统的初始状态对仿真结果的影响可忽略。然而，为了得到系统稳态性能的较好的估计值，需要确定仿真运行长度到底多长就可以认为是“足够”了。

例 10.3 考虑估计 M/M/1 的稳态排队等待时间 d，它应满足

$$d=\lim_{m\to\infty}\sum_{i=1}^{m} D_i/m \qquad (l(0)=\text{任意 } l) \tag{10.3}$$

实际中，m 不可能无穷大，需要讨论如何由一次有限 m 的仿真运行中得到 d 的好的估计值。

■

10.2 终止型仿真输出分析

终止型仿真的要求是每次运行的初始条件相同，但必须是相互独立的。实现独立运行的方法是每次采用不同的随机数据流。

如果 X_i 是第 i 次运行时得到的某一系统性能的仿真结果(如例 10.2 中的 D_i)，由于每次运行相互独立，则可以认为 X_i 是独立同分布的随机变量，从而可以用经典统计分析的方法构造 $E(X)$的置信区间。

根据对置信区间的精度要求，终止型仿真结果分析有两类基本方法，即固定样本长度法及终止型序贯程序法，下面我们讨论这两种方法。

10.2.1 固定样本长度法

由于每次仿真运行得到的结果是系统性能的一个样本，固定样本长度法实际上就是由用户规定独立运行的次数 $n(n\geqslant 2)$。

假定每次运行的结果 $X_1, X_2, \cdots, X_n$ 除了满足独立同分布的条件外，而且是正态随机量，则随机变量 X 的期望值 $E(X)$的估计值 μ 为

$$\mu = \overline{X}(n) \pm t_{n-1,1-\alpha/2}\sqrt{S^2(n)/n} \tag{10.4}$$

其中，$\overline{X}(n)=\sum_{i=1}^{m} X_i/n$，$\alpha$ 称为置信水平，而

$$S^2(n) = \sum_{j=1}^{m}[\overline{X}(n) - X_j]^2/(n-1) \tag{10.5}$$

例 10.4 对例 10.1 的 M/M/1 系统独立运行 10 次，每次长度为 200，初始条件是初始队长 $q(0)=0$，服务台状态为闲，仿真运行的结果如表 10.2 所示。

表 10.2 M/M/1 系统终止型仿真结果

j	1	2	3	4	5	6	7	8	9	10
平均等待时间 $D_j(200)$	10.427	14.469	12.780	8.703	12.727	9.206	8.053	28.039	6.228	13.931
平均队长 $Q_j(200)$	2.098	2.718	2.389	1.596	2.585	1.755	1.724	6.523	1.227	2.679

可以得到

$$\overline{D}(10) = \sum_{j=1}^{10}\overline{D}_j(200)/10 = 12.456$$

$$\overline{Q}(10) = \sum_{j=1}^{10}\overline{Q}_j(200)/10 = 2.539$$

$$S_D^2(10) = \sum_{j=1}^{10}[\overline{D}(10) - \overline{D}_j(200)]^2/9 = 37.27$$

$$S_Q^2(10) = \sum_{j=1}^{10}[\overline{Q}(10) - \overline{Q}_j(200)]^2/9 = 2.296$$

从而可得到平均排队等待时间的期望值 $d(200,l(0)=0)$ 及平均队长的期望值 $q(200,l(0)=0)$ 在 $\alpha=0.10$ 时的估计值为

$$\overline{D}(10)\pm t_{9,0.95}\sqrt{S_D^2(10)/10}=12.456\pm 3.537$$

$$\overline{Q}(10)\pm t_{9,0.95}\sqrt{S_Q^2(10)/10}=2.539\pm 0.878$$

因而，我们可以相信，$d(200,l(0)=0)$ 以将近 90% 的置信度位于区间[8.919,15.993]上，$q(200,l(0)=0)$ 以将近 90% 的置信度位于区间[1.661,3.417]上。 ■

显然，固定样本长度法得到的估计值依赖于 X_j 是正态随机变量这一假设。根据中心极限定理，若产生 X_j 的样本点数越多，即每次仿真运行的次数越多，则 X_j 越接近正态分布。因此，在终止型仿真中，每次仿真运行的次数不能太少。否则 X_j 的分布可能不对称而造成歪斜，因而由 X_j 建立的置信区间覆盖真值的程度将会降低。

固定样本长度法所得到的置信区间长度不但与 X_j 的方差有关，而且与仿真运行次数有关。在这种方法中，X_j 的方差主要决定于仿真模型，而仿真运行次数则是由用户规定的，为了减少置信区间的长度，显然需要加大 n，因为置信区间长度与$\sqrt{n}$成反比。根据这一特点，我们可以得到有规定精度的置信区间，这就是基于固定样本长度法的序贯法。

10.2.2 终止型序贯程序法

固定样本长度法对构造的置信区间长度未加控制。如例 10.4 中所得到的顾客排队等待时间的置信区间为[8.919,15.993]，半长为 3.537，对某些研究来说，这个长度可能太大了。

由于置信区间的长度反比于$\sqrt{n}$，那么可以通过增加 n 来减少区间的长度以提高精度。置信区间的半长称为它的绝对精度，用 β 表示，而将置信区间半长与点估计的绝对值之比称为置信区间的相对精度，用 γ 表示。

为了得到规定的 β,γ，可先按固定样本长度法运行 n 次，若得到的 β 或 γ 太大，可再增加 n，一种解析地确定 n 的做法是，设 X_j 的总方差估计 $S^2(n)$ 随着 n 加大而没有显著变化，则

$$n_a(\beta)=\min\left\{i\geqslant n:\ t_{i\ 1,1-\alpha/2}\sqrt{\frac{S^2(n)}{i}}\leqslant\beta\right\} \tag{10.6}$$

或

$$n_a(\gamma)=\min\left\{i\geqslant n:\ t_{i-1,1\ \alpha/2}\sqrt{\frac{S^2(n)}{i}}\Big/|\ \overline{X}(n)\ |\leqslant\gamma\right\} \tag{10.7}$$

注意，在式(10.7)中还假设 $\overline{X}(n)$ 随 n 加大也没有明显变化，即认为 $\overline{X}(i)=\overline{X}(n)(i\geqslant n)$。

例 10.5 设 M/M/1 系统 10 次运行得到的顾客排队平均等待时间(采用固定样本长度法)如表 10.3 所示。

表 10.3 M/M/1 系统固定样本长度法仿真结果

j	1	2	3	4	5	6	7	8	9	10
D_j	1.051	6.438	2.646	0.805	1.505	0.546	2.281	2.822	0.414	1.307

由上述 10 次运行得到的总均值及总方差分别为

$$\overline{D}(10)=1.982,\quad S_D^2(10)=3.172$$

现在要求在 $\alpha=0.10$ 下构造置信区间，使其绝对精度 $\beta=0.5$，相对精度 $\gamma=0.15$，按式(10.6)及式(10.7)可解析地计算出

$$n_{0.10}(0.5)=\min\left\{i\geqslant 10:\ t_{i-1,0.95}\sqrt{\frac{3.172}{i}}\leqslant 0.5\right\}=37$$

$$n_{0.10}(0.15)=\min\left\{i\geqslant 10:\ t_{i-1,0.95}\sqrt{\frac{3.172}{i}}/1.982\leqslant 0.15\right\}=99$$ ■

实验表明，随着 n 的加大，认为 $S^2(n)$ 保持不变的条件过于苛刻，从而按式(10.6)或式(10.7)计算得到的 $n_\alpha(\beta)$ 或 $n_\alpha(\gamma)$ 偏大，因而往往采用序贯程序法，这种方法的步骤如下：

(1) 预定独立仿真运行的次数 $n_0\geqslant 2$，并置 $n=n_0$，独立运行 n 次；

(2) 计算该 n 次运行的 $X_1,X_2,\cdots,X_n$，以及相应的 $\overline{X}(n)$ 及 $S^2(n)$；

(3) 按式(10.6)或式(10.7)计算出 β_n 或 γ_n；

(4) 若 $\beta_n\leqslant\beta$ 或 $\gamma_n\leqslant\gamma$，则置信区间为

$$[\overline{X}(n)-\beta_n,\overline{X}(n)+\beta_n]$$

并将其作为 μ 在近似 $100(1-\alpha)\%$ 意义下的置信区间，从而结束仿真，否则

(5) 再进行一次独立的仿真运行得到 X_{n+1}；

(6) 令 $n=n+1$，并返回(2)。

采用上述序贯程序法，在同样精度的要求下，仿真运行的次数较之解析得到的次数要少得多。如例 10.5 中 $n_{0.10}(0.15)=99$，而采用序贯程序法得到的 $n=68$，$\overline{X}(68)=1.872$，$S^2(68)=1.865$，可见 $S^2(68)$ 较 $S^2(10)$ 要小得多。

10.3 稳态型仿真输出分析

仿真研究的另一种常见的情况是估计的稳态性能，此时，一般是执行一次长度很大的仿真运行。

令 $Y_1,Y_2,\cdots,Y_m$ 是从某次运行得到的输出过程，则 $\{Y_i,i\geqslant 1\}$ 的稳态平均响应 ν 由下式定义

$$\nu=\lim_{m\to\infty}\sum_{i=1}^{m}Y_i/m \tag{10.8}$$

式(10.8)与式(10.3)的形式完全相同，但更具有一般性。当然，这要求 ν 的极值存在，这样 ν 与仿真的初值无关。

由仿真结果 $\{Y_i,i\geqslant 1\}$ 估计 ν 的置信区间在稳态型仿真中主要有两种方法，一种称为批均值法，另一种是稳态型序贯程序法。另外我们介绍两种基于不同思想的估计稳态性能的方法，即重复产生法及重复删除法。

10.3.1 批均值法

批均值法的基本思想是：设仿真运行长度为 m(m 足够大)，则得到输出过程 $Y_1,Y_2,\cdots,Y_m$，将 $\{Y_i,i=1,2,\cdots,m\}$ 分为 n 批，每批长度为 l，则得到每批数据如下

$$\begin{array}{cccc} Y_1 & Y_2 & \cdots & Y_l \\ Y_{l+1} & Y_{l+2} & \cdots & Y_{2l} \\ \vdots & \vdots & & \vdots \\ Y_{(n-1)l+1} & Y_{(n-1)l+2} & \cdots & Y_{nl} \end{array}$$

分别对每批数据进行处理，求得每批的均值为 $\bar{Y}_j(l)(j=1,2,\cdots,n)$，则总的样本均值为

$$\bar{Y}(n,l) = \sum_{j=1}^{n} \bar{Y}_j(l)/n = \sum_{i=1}^{m} Y_i/m$$

我们将 $\bar{Y}(n,l)$ 作为 ν 的点估计。

为了构造 ν 的置信区间，必须对 $\bar{Y}_j(l)$ 有一定要求，若 $\bar{Y}_j(l)$ 是独立的且服从正态分布的随机变量，并具有相同的均值与方差，则 ν 的近似 $100(1-\alpha)\%$ 的置信区间的计算公式为

$$\bar{Y}(n,l) \pm t_{n-1,1-\alpha/2} \sqrt{S^2_{\bar{Y}_j(l)}(n)/n} \tag{10.9}$$

$$S^2_{\bar{Y}_j(l)}(n) = \sum_{l=1}^{n} [\bar{Y}_j(l) - \bar{Y}(n,l)]^2/(n-1) \tag{10.10}$$

例 10.6 对例 10.1 的 M/M/1 系统，设每批长度 $l=1\,000$，共 20 批，则仿真运行长度 $m=20\,000$，经仿真后得出，系统平均排队等待时间为 15.84min，与理论值 16.00 相差 0.16min，相对误差为 1%，平均队长为 3.203，与理论值相差 0.003，相对误差为 0.9%。再根据其方差确定其置信区间，设 $\alpha=0.10$，经查 t 分布表可得 $t_{19,0.95}=1.729$，从而

$$t_{19,0.95} \sqrt{S^{2(20)}_{D_j(1\,000)}/20} = 2.001$$

$$t_{19,0.95} \sqrt{S^{2(20)}_{Q_j(1\,000)}/20} = 0.421$$

这样，可以有近似 90% 的把握确信 d 及 q 的置信区间分别为[13.839,17.84]及[2.782,3.624]。 ■

为了提高批均值法的有效性，$\bar{Y}_i(l)$ 必须满足前述的条件，那么，怎样才能满足这些条件呢？首先，为使 $\bar{Y}_i(l)$ 独立，则要求每批长度 l 要足够大。l 到底要多大，不同系统的要求是不同的，在下一节我们讨论一种确定 l 的方法。其次，为使 $\bar{Y}_i(l)$ 接近正态分布，除了要求 l 足够大外，且批数 n 也要足够大。由此可以看到，为使 $\bar{Y}_i(l)$ 独立且服从正态分布，则要求 $m=nl$ 足够大，这也就是稳态仿真的定义所必须满足的条件。

另外，$\bar{Y}_i(l)$ 应具有相同的均值与相同的方差，则要求 $Y_1,Y_2,\cdots,Y_m$ 是协方差平稳过程。

采用批均值法构造置信区间的原理是比较简单的，但在实际使用中常常会出现偏差，其原因是上述 $\bar{Y}_i(l)$ 的要求得不到满足。如果 l 不够大，$\bar{Y}_i(l)$ 具有很强的相关性，因而 $S^2_{\bar{Y}_j(l)}(n)/n$ 严重偏离 $\sigma^2[\bar{Y}(n,l)]$ 的估计值；特别是，如果 $\bar{Y}_i(l)$ 是正相关的，由 $\bar{Y}_i(l)$ 得到的方差估计偏低得特别厉害，从而置信区间偏小，甚至有可能覆盖不了真值。造成偏差的另一个原因则是 $Y_1,Y_2,\cdots,Y_m$ 不满足协方差平稳过程这一条件。

例 10.6 的仿真十分接近理论值的原因在于 $\bar{Y}_i(l)$ 及 $Y_1,Y_2,\cdots,Y_m$ 满足上述条件的程度较好，每批长度 l 足够大，批数 n 也足够多(对该模型而言)。实际上，若取 $l=500$，同样进行 20 批统计，该次仿真运行的平均排队等待时间 $\bar{D}(20,500)=17.627$，平均队长 $\bar{Q}(20,500)=3.581$，$S^2_{D_j(500)}(20)=82.422\,02$，$S^2_{Q_j(500)}(20)=3.623\,75$，其 d 的相对误差为 10.2%，q 的相对误差为 11.9%，置信区间的半长分别为 3.51 及 0.736，分别是 $l=1\,000$ 时的将近 2 倍，可

见 l 的选择是十分重要的。

批均值法对置信区间的精度未加控制，下面我们讨论基于批均值法的稳态型序贯程序法，以满足规定精度置信区间的要求。

10.3.2 稳态型序贯法

设某次稳态运行得到的观测值是 $Y_1, Y_2, \cdots, Y_m$，其批长度为 l，共 n 批，每批均值为 $\bar{Y}_j(l)(j=1,2,\cdots,n)$，总体样本均值为 $\bar{\bar{Y}}(n,l)$。

记任意两批之间的相关系数为 $\rho_i(l)$，则

$$\rho_i(l) = Cov[\bar{Y}_j(l), \bar{Y}_{j+i}(l)](i = 1,2,\cdots,n-1)$$

又令

$$b(n,l) = E\{\hat{\sigma}^2[\bar{\bar{Y}}(n,l)]\}/\sigma^2[\bar{\bar{Y}}(n,l)]$$

其中

$$\hat{\sigma}^2[\bar{\bar{Y}}(n,l)] = S^2_{\bar{Y}_j(l)}(n)/n$$

若 $Y_1, Y_2, \cdots$ 是协方差平稳过程，则可以证明，当 $l \to \infty$ 时，$\rho_i(l) \to 1$。如果在批均值的基础上，根据相关数 $\rho_i(l)$ 的大小来确定 l，使得 $\rho_i(l)$ 达到足够小。此时，可认为 $\bar{Y}_i(l)$ 基本上是独立的或接近独立的。

为得到不相关的 $\bar{Y}_i(l)$，直观的做法是，批数 n 不变，然后不断地加长 l，直到 $\rho_i(l)$ 的估计值小于规定值为止。但是，如果 n 选择过小，则其方差加大，从而 $b(n,l)$ 将远小于 1，结果得到的置信区间就会偏大，为此 n 也要加大。因此，为达到一定精度的 $\rho_i(l)$ 且使 $b(n,l)$ 接近 1，则要求 $m=nl$ 就会特别大。

综合考虑 $\rho_i(l)$ 与 $b(n,l)$ 两方面的要求，下面我们介绍一种尽可能减少 m 的方法。

这种方法的基本思想是，设某次仿真运行的批长度为 l，共 fn 批($f \geqslant 2$)，根据相隔 i 批的相关系数 $\rho_i(l)$ 来推断 n 批长度为 fl 的数据是否接近不相关。

这种方法的关键是 $\rho_i(l)$ 与 $\rho_i(fl)$ 之间的关系是否有规律可循。实验表明，$\rho_i(l)$ 随 l 变化的规律大致有以下三种情况：

(1) $\rho_i(l)$ 为递减函数。

对这一类系统，实验数据是，如果 $n=40$，$f=10$，对于某个 l 若能使 $\rho_i(l)<0.4$，则 $\rho_i(10l)<0.05$，且 $0.9<b(40,10l)<1$(参见图 10.1)。

(2) $\rho_i(l)$ 的值一次或多次改变方向，然后严格地减少到 0(参见图 10.2)。

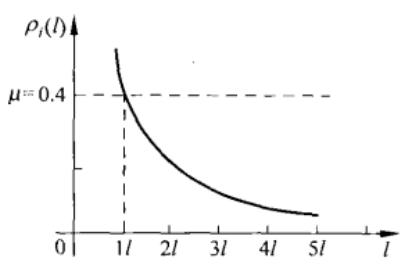


图 10.1 $\rho_i(l)$ 为单调递减函数

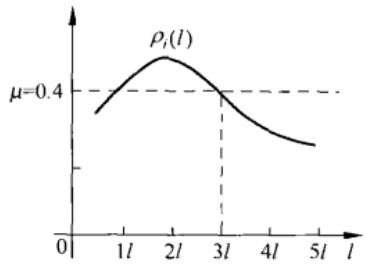


图 10.2 $\rho_i(l)$ 多次改变方向然后递减

这类系统的实验数据是，如果 $\rho_i(l)<0.4$，$f=10$，$n=40$，且当 $l'>l$ 满足 $\rho_i(l')<\rho_i(l)$，那么 $\rho_i(10l)<0.05$，且 $0.9<b(40,10l)<1$。

(3) $\rho_i(l)<0$ 或者随 l 变化无一定规律。

在这种情况下显然无法由 $\rho_i(l)$ 推断 $\rho_i(10l)$，但是至少可以保证，由 $\bar{Y}_i(10l)$ 得到的置信区间不比由 $\bar{Y}_i(l)$ 得到的差。

根据 $\rho_i(l)$ 的上述三种特性，基于批均值法的稳态型序贯法原理如下(参见图 10.3)：

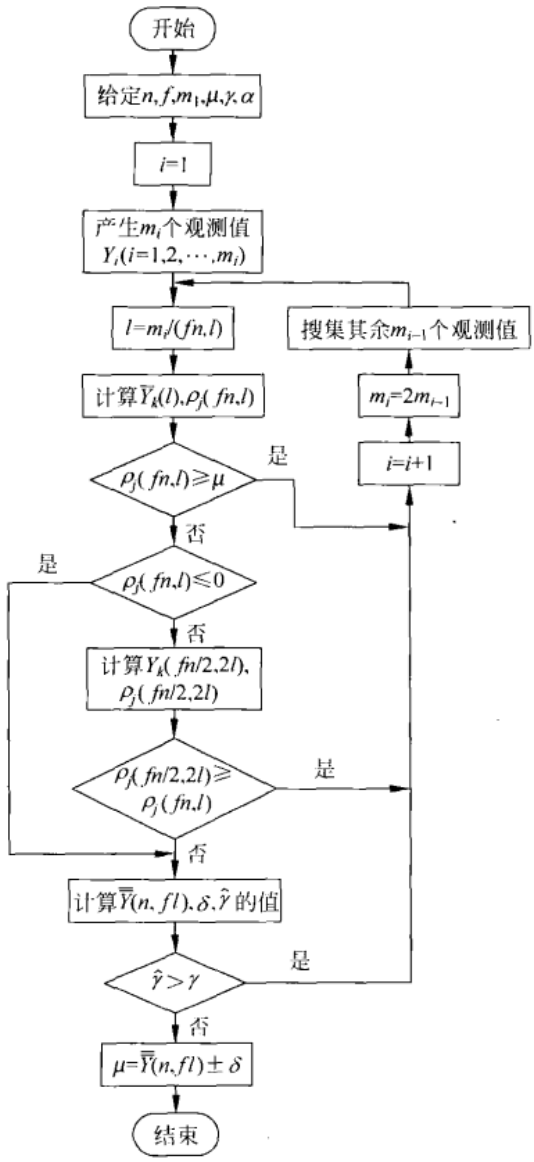


图 10.3 稳态序贯法流程图

(1) 给定批数因子 n，f 及仿真长度 m_1(m_i 是 nf 的整数倍)，$\rho_i(l)$ 的判断值 μ，置信区间的相对精度 γ，置信水平 α，并令 $i=1$。

(2) 运行长度为 m_i 的仿真，得到 m_i 个观测值 $Y_1,Y_2,\cdots,Y_{m_i}$。

(3) 令 $l=m_i/(fn)$，计算 $\bar{Y}_K(l)(K=1,2,\cdots,fn)$ 及 $\rho_j(fn,l)$(一般取 $j=1$ 即滞后 1 的

相关系数)。

(4) 若 $\rho_i(fn,l)\geqslant\mu$,则说明 m_i 太小即运行长度太短,需加大 m_i,可令 $i=i+1$,且 $m_i=2m_{i-1}$,在第(2)步的基础上再进行长度为 m_{i-1} 的仿真,得到其余 m_{i-1} 个观测值,然后返回第(3)步。

(5) 若 $\rho_i(fn,l)\leqslant 0$,则说明是第 3 类情况的相关系数,增加仿真长度无助于 $\rho_i(l)$ 的判断,直接到第(8)步。

(6) 若 $0<\rho_i(fn,l)<\mu$,计算 $Y_K(fn/2,2l)(K=1,2,\cdots,fn/2)$ 及 $\rho_i(fn/2,2l)(i=1)$,判断 $\rho_i(l)$ 是否具有第 2 类特征。

若 $\rho_i(fn/2,2l)\geqslant\rho_j(fn,l)$,则说明该 $\rho_i(l)$ 的确具有第 2 类特征,需进一步加大 m_i,同样令 $i=i+1$,$m_i=2m_{i-1}$,在第(2)步的基础上再继续运行长度为 m_{i-1} 的仿真,得到新的 m_{i-1} 个观测值,然后返回第(3)步。

(7) 若 $\rho_i(fn/2,2l)<\rho_i(fn,l)$,则说明 ρ_j 已具有第 1 类特征,且到达 $\rho_i(l)$ 判断值 u 的 l 已得到,那么可以相信 $\rho_i(n,fl)$ 的值满足独立性的要求,此时用批均值法计算该 n 批,长度为 fl 的置信区间,即进入第(8)步。

(8) 计算 $\overline{Y}_K(n,fl)$,$\overline{\overline{Y}}(n,fl)$ 以及置信区间的半长 $\delta=t_{n-1,1-\alpha/2}\sqrt{S^2_{\overline{Y}(n,fl)}(n)/n}$,最后得到 $\hat{\gamma}=\delta/\overline{\overline{Y}}(n,fl)$。

(9) 如果 $\hat{\gamma}>\gamma$,说明精度不满足要求,要继续加大仿真运行长度。令 $i=i+1$,$m_i=2m_{i-1}$,在第 2 类的基础上再继续运行长度为 m_{i-1} 的仿真,得到新的 m_{i-1} 个观测值,然后回到第(3)步。

(10) 如果 $\hat{\gamma}\leqslant\gamma$,则达到精度要求,可令估计值 $\mu=\overline{\overline{Y}}(n,fl)\pm\delta$,然后停止仿真。

稳态序贯法较好地解决了批长度的确定、仿真运行总长度的确定问题,并能满足规定的置信区间精度的要求。实际使用时,一般可选择 $n=40$,$f=10$,$\mu=0.4$,$\gamma=0.075$,m_1 则可根据系统的情况进行初选。实践表明,这种方法的效果比较理想。

值得说明的是,上述讨论是在已知 $\rho_i(l)$ 的条件下进行的。实际上,在仿真中,$\rho_i(l)$ 是其估计值 $\hat{\rho}_i(l)$,即

$$\hat{\rho}_i(l)=\frac{\sum_{K=1}^{n-1}[\overline{Y}_K(l)-\overline{\overline{Y}}(n,l)][\overline{Y}_{K+j}(l)-\overline{\overline{Y}}(n,l)]}{\sum_{K=1}^{n}[\overline{Y}_K(l)-\overline{\overline{Y}}(n,l)]^2} \tag{10.11}$$

当 $j=1$ 时

$$\hat{\rho}_i(l)=\frac{\sum_{K=1}^{n-1}[\overline{Y}_K(l)-\overline{\overline{Y}}(n,l)][\overline{Y}_{K+1}(l)-\overline{\overline{Y}}(n,l)]}{\sum_{K=1}^{n}[\overline{Y}_K(l)-\overline{\overline{Y}}(n,l)]^2} \tag{10.12}$$

实验表明,如果 $\hat{\rho}_1^1(n/2l)$ 与 $\hat{\rho}_1^2(n/2l)$ 分别是基于前 $n/2$ 批与后 $n/2$ 批(设 n 为偶数)的滞后 1 的相关系数的估计值,则采用式(10.13)修正 n 批的 $\hat{\rho}_1(l)$ 后其偏差会更小些:

$$\hat{\rho}_1(n,l)=2\hat{\rho}_1(l)-\frac{\hat{\rho}_1^1(n/2,l)+\hat{\rho}_1^2(n/2,l)}{2} \tag{10.13}$$

10.3.3 重新产生法

批均值法是将一次长度为 m 的稳态型仿真分成等长的若干批，然后分批进行数据处理。在这种方法中，批长度的确定是十分重要的，它对批均值法的效能有直接的影响，然而在这方面尚没有十分有效的原则可循。

重新产生法的基本思想是，在一次稳态型仿真中，设仿真从某一初始状态开始，当运行到系统重新达到该状态时，其以后的过程可以认为是与前面的过程独立的，这一过程称为重新产生周期，设 N_j 是第 j 个周期样本点$\{Y_i, i\geqslant 1\}$的个数，Z_j 是该 N_j 个样本之和，即

$$Z_j = \sum_{K=1}^{N_j} Y_{jK} \tag{10.14}$$

其中 Y_{jK} 表示仿真输出过程$\{Y_i, i\geqslant 1\}$在第 j 个周期上的第 K 个数据$(1\leqslant K\leqslant N_j)$，若 $E(N_j)<\infty$，$E(|Z_j|)<\infty$，则该系统的稳态平均响应 ν 可由下式确定

$$\nu = E(Z)/E(N) \tag{10.15}$$

在实际仿真中，我们是用样本均值代替期望值，即用

$$\hat{\nu}_J = \bar{Z}_J/\bar{N}_J \tag{10.16}$$

作为 ν 的估计值，其中 J 为重新产生的周期数。

这种方法初看起来似乎不好理解，事实上，由于

$$Z_j = \sum_{K=1}^{N_j} Y_{jK} \quad (j = 1,2,\cdots,J)$$

并令 $N = \sum_{j=1}^{J} N_j$，则

$$\hat{\nu}_J = \frac{\sum_{j=1}^{J}\sum_{K=1}^{N_j} Y_{jK}}{N} = \frac{\sum_{j=1}^{J}\sum_{K=1}^{N_j} Y_{jK}}{\sum_{j=1}^{J} N_j} = \frac{\sum_{j=1}^{J} Z_j}{\sum_{j=1}^{J} N_j} = \frac{\sum_{j=1}^{J} Z_j/J}{\sum_{j=1}^{J} N_j/J} = \bar{Z}_J/\bar{N}_J$$

虽然 $\bar{Z}_J$ 及 $\bar{N}_J$ 分别是 $E(Z)$及 $E(N)$的无偏估计，但 $\hat{\nu}_J$ 却不是 ν 的无偏估计，但可以证明，$\hat{\nu}_J$ 确是 ν 的强一致估计，即当 $J\to\infty$时，$\hat{\nu}_J$ 趋近 ν 的概率为 1。

例 10.7 考虑 M/M/1 系统，要求估计系统的稳态平均排队等待时间 d 及稳态平均队长 q。

设仿真的初始状态为：队长为 0，服务台空闲，那么，当一个新的顾客到达时队长为 0 且服务台为空闲时为重新产生周期的第 1 个样本点(亦称为重新产生点)。设每个顾客在队列中排队的等待时间为$\{D_i, i\geqslant 1\}$，队长为$\{Q(t), i\geqslant 0\}$，则按稳态仿真的定义有

$$d = \lim_{m\to\infty}\sum_{i=1}^{N} D_i/m$$

$$q = \lim_{t\to\infty}\int_0^t Q(u)\mathrm{d}u/t$$

其中，m 是稳态仿真运行的长度。那么，按重新产生法得

$$Z_j = \sum_{i=1}^{N_j} D_i, \quad Y_j = \int_{t_{j-1}}^{t_j} Q(u)\mathrm{d}u$$

其中，N_j 是第 j 个周期内排队等待时间的观测点数，而 $T_j = t_j - t_{j-1}$ $(j=1,2,\cdots,J)$ 为第 j 个周期的仿真时间长度，则用

$$\hat{d} = \bar{Z}_J / \bar{N}_J, \quad \hat{q} = \bar{Y}_J / \bar{T}_J$$

作为 d,q 的稳态估计值。 ■

既然 $\hat{\nu}$ 对 ν 的估计是有偏的，那么如何确定其置信区间呢？

令 $\boldsymbol{U}_j = (Z_j, N_j)^{\mathrm{T}}$ 的协方差矩阵为

$$\sum = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix} \tag{10.17}$$

其中

$$\sigma_{11} = E\{[Z_j - E(Z_j)][Z_j - E(Z_j)]\} \tag{10.18}$$

$$\sigma_{12} = E\{[Z_j - E(Z_j)][N_j - E(N_j)]\} \tag{10.19}$$

$$\sigma_{22} = E\{[N_j - E(N_j)][N_j - E(N_j)]\} \tag{10.20}$$

若令

$$V_j = Z_j - \nu N_j \tag{10.21}$$

则 V_j 是均值为 0，方差为

$$\sigma_V^2 = \sigma_{11} - 2\nu\sigma_{12} + \nu^2\sigma_{22} \tag{10.22}$$

的独立同分布的随机变量，根据中心极限定量，当 $J \to \infty$ 时

$$\frac{\bar{V}_J}{\sqrt{\sigma_V^2 / J}} \to \mathrm{N}(0,1) \tag{10.23}$$

其中，$\to$ 表示收敛于某种分布。实际仿真中，只能得到 $\boldsymbol{U}_j$ 的估计的协方差矩阵

$$\hat{\sum}_J = \begin{bmatrix} \hat{\sigma}_{11} & \hat{\sigma}_{12} \\ \hat{\sigma}_{12} & \hat{\sigma}_{22} \end{bmatrix} = \frac{\sum_{j=1}^{J} (\boldsymbol{U}_j - \bar{\boldsymbol{U}}_j)(\boldsymbol{U}_j - \bar{\boldsymbol{U}}_j)^{\mathrm{T}}}{J-1} \tag{10.24}$$

则 V_j 的方差估计值为

$$\hat{\sigma}_V^2(J) = \hat{\sigma}_{11} - 2\nu_j\hat{\sigma}_{12} + \nu^2\hat{\sigma}_{22}$$

由于 $\bar{V}_J = \bar{Z}_J - \nu\bar{N}_J$，则

$$\frac{\bar{V}_J}{\sqrt{\hat{\sigma}_V^2(J)/J}} = \frac{\bar{Z}_J - \nu\bar{N}_J}{\sqrt{\hat{\sigma}_V^2(J)/J}} \to \mathrm{N}(0,1)$$

从而

$$\frac{\bar{Z}_J/\bar{N}_J - \nu}{\sqrt{\hat{\sigma}_V^2(J)/[J(\bar{N}_J)^2]}} = \frac{\hat{\nu}_J - \nu}{\sqrt{\hat{\sigma}_V^2(J)/[J(\bar{N}_J)^2]}} \to \mathrm{N}(0,1)$$

那么，如果 J 足够大，ν 的近似 $100(1-\alpha)\%$ 置信区间为

$$\hat{\nu}_J \pm \frac{Z_{1-\alpha/2}\sqrt{\hat{\sigma}_V^2(J)/J}}{\bar{N}_J} \tag{10.25}$$

重新产生法的缺点在于其重新产生点数量要足够多，且每个周期应是独立的，而实际系

统可能没有重新产生点或周期太长。重新产生法的效率正比于 $\sum_{j=1}^{J} N_j$。若 N_j 太大，则要求运行的总长度要足够大。另外，这种方法难以预先确定置信区间的精度，因而无法得到规定精度的置信区间。

10.3.4 重复删除法

前面分别讨论了终止型仿真及稳态型仿真结果分析方法。人们可能要问，能否由终止型仿真结果来估计系统稳态响应呢？例如，假如终止型仿真运行次数 K 趋向无穷大时，能否认为下式

$$d(m \mid l(0)=0)=\lim_{K\to\infty}\left[\frac{\sum_{j=1}^{K}\bar{D}_j}{K} \mid l(0)=0\right]$$

得到的 M/M/1 系统的 d 就是系统的稳态平均排队等待时间呢？结论是否定的。

例如，曾对例 10.1 的系统进行过 20 次独立终止型仿真，每次运行长度为服务 200 个顾客，初始条件相同，采用独立的随机数，这 20 次运行结果如表 10.4 所示。

表 10.4 M/M/1 系统固定样本长度法 20 次仿真结果

运行编号	平均延误时间	平均队长	运行编号	平均延误时间	平均队长
1	10.427	2.098	11	4.940	0.939
2	10.822	2.546	12	17.505	3.449
3	9.255	1.755	13	12.991	2.778
4	11.245	1.952	14	9.112	1.858
5	29.497	6.268	15	14.982	2.952
6	10.678	2.298	16	15.555	3.134
7	12.375	2.766	17	7.919	1.446
8	12.327	2.285	18	15.707	3.398
9	14.469	2.718	19	10.356	2.004
10	12.780	2.389	20	11.389	2.372

用终止型固定样本长度法来确定其总均值及 $\alpha=0.10$ 时置信区间的半长，其数据如表 10.5 所示。

表 10.5 M/M/1 系统固定样本长度法置信区间

运行次数 K	平均延误时间	平均队长	平均延误时间 置信区间半长	平均队长 置信区间半长
5	14.249 2	2.923 9	8.157 8	1.804 0
10	13.387 5	2.707 5	3.388 5	0.748 4
15	12.893 7	2.603 4	2.470 2	0.533 9
20	12.716 6	2.570 3	1.904 4	0.415 5

显然，随着 K 的加大，其总均值 $\bar{D}$ 及 $\bar{Q}$ 并未越来越接近稳态理论值，反而越来越偏离稳态理论值（$\bar{D}$ 的稳态理论值为 16.0，$\bar{Q}$ 的稳态理论值为 3.2），而其置信区间的半长却越来

越小。可以预料，当 K 趋向无穷大时，由于区间半长

$$t_{K\ 1.1\ \alpha/2}\sqrt{S^2(K)/K}$$

将趋向 0，而其总均值却未必能接近稳态值。

造成这种结果的原因在于，终止型仿真时，虽然每次运行是独立的，但其系统的初始状态是完全相同的，而这种初始状态不一定能代表系统稳定特性的状态。由于每次仿真长度有限，初始状态对仿真结果的影响未消除，所得到的结果必然是系统的有偏估计。

终止型仿真这种初始状态的影响，在仿真文献中称之为启动问题。为了克服这种初始状态对系统性能的影响，以便由多次终止型仿真来估计系统的稳态性能，人们提出了所谓重复删除法。

这种方法的基本思想是：设对某一系统进行 K 次独立的终止型仿真，每次长度为 m，得到下列观测值

$$\begin{aligned}&Y_{11},Y_{12},\cdots,Y_{1m}\\&Y_{21},Y_{22},\cdots,Y_{2m}\\&\vdots\\&Y_{K1},Y_{K2},\cdots,Y_{Km}\end{aligned}$$

则 $Y_{ji}(j=1,2,\cdots,K,i=1,2,\cdots,m)$是第 j 次运行得到的第 i 个观测值。在统计计算系统性能时，删除每一次运行的前 l 个观测数据，其中 $0\leqslant l\leqslant m$，并令

$$\bar{Y}_j(m,l)=\sum_{i=l+1}^{m}Y_{ji}/(m-l) \tag{10.26}$$

$$\bar{\bar{Y}}(K,m,l)=\sum_{i=l\,1}^{K}Y_j/(m-l)/K \tag{10.27}$$

将 $\bar{Y}_j(m,l)$作为每次运行的均值，$\bar{\bar{Y}}(K,m,l)$作为系统稳态性能的估计值，其置信区间半长为

$$t_{K\ 1,1-\alpha/2}\sqrt{S^2_{Y_j(m,l)}(K)/K} \tag{10.28}$$

其中

$$S^2_{Y_j(m,l)}(K)=\frac{1}{K-1}\sum_{i=1}^{K}(\bar{Y}_j(m,l)-\bar{\bar{Y}}(K,m,l))^2 \tag{10.29}$$

重复删除法显然具有很强的吸引力。它只需要运行 K 次独立的终止型仿真，所需样本容量可以大大地减少，问题是如何确定 l 的值。

考虑到系统稳态运行时某一时刻系统的状态仍具有随机性，那么，如果用一个独立性及均匀性较好的随机数发生器产生[0,m]区间内的随机整数 l，在每次仿真运行时，用该 l 来控制删除的观测数据个数，则可以认为所得到的 $\bar{Y}_j(m,l)$是独立同分布的随机变量，由此 $\bar{Y}_j(m,l)$得到的 $\bar{\bar{Y}}(K,m,l)$是系统稳态性能的无偏估计。

我们仍然以例 10.1 的系统应用重复删除法。前面我们已经看到，若每次运行长度为 200，分别独立运行 5 次，10 次，15 次及 20 次的结果，由这些仿真结果得到的估计值是系统的有偏估计。表 10.6 是应用重复删除法的结果（表中 L 是每次删除的被服务的样本数，$\bar{D}$ 是随机删除后剩余样本的平均延误时间，$\bar{Q}$ 是随机删除后剩余样本的平均队长）。

表 10.6 M/M/1 系统重复删除法仿真结果

运行次数序号	L	$\bar{D}$	$\bar{Q}$	运行次数序号	L	$\bar{D}$	$\bar{Q}$
1	3	13.658	2.568	21	77	2.620	0.667
2	14	10.028	2.161	22	65	8.888	1.947
3	66	9.452	1.836	23	21	11.368	2.418
4	88	24.047	5.565	24	77	43.902	13.261
5	28	35.839	7.329	25	54	10.797	1.478
6	96	1.077	1.309	26	1	12.641	2.779
7	35	4.853	1.059	27	48	2.977	0.660
8	10	10.700	2.039	28	64	4.047	0.906
9	10	22.315	4.539	29	6	7.876	1.527
10	11	22.348	4.456	30	23	9.738	1.622
11	15	14.526	2.909	31	32	8.123	1.655
12	86	30.434	7.361	32	29	28.593	8.526
13	75	14.455	2.346	33	66	10.589	3.171
14	63	16.203	3.667	34	24	11.607	2.363
15	86	7.008	1.302	35	14	39.415	9.407
16	10	10.101	1.950	36	33	18.748	4.190
17	35	11.057	1.941	37	7	9.468	1.981
18	72	31.991	8.754	38	46	5.002	0.859
19	74	12.834	3.170	39	52	11.226	3.509
20	23	6.516	1.316	40	14	10.830	2.048

用上述随机重复删除法的仿真结果，分别统计 5 次，10 次，15 次，…，40 次的结果来确定其总均值及 $\alpha=0.10$ 时置信区间的半长，其数据如表 10.7 所示。

表 10.7 M/M/1 系统重复删除法置信区间

运行次数 K	平均延误时间	平均队长	平均延误时间 置信区间半长	平均队长 置信区间半长
5	16.876 8	3.892 0	12.997 4	2.315 2
10	16.255 4	3.675 4	6.759 7	1.263 0
15	16.303 9	3.608 2	4.732 0	0.968 2
20	15.855 3	3.559 3	3.882 8	0.889 1
25	15.769 3	3.636 3	3.806 2	1.016 5
30	14.380 8	3.282 6	3.298 0	0.876 8
35	15.096 1	3.521 4	3.163 7	0.848 0
40	14.608 8	3.401 8	2.806 1	0.751 7

从上面的结果我们可以看到，虽然每次独立运行长度为 100(是前面运行的一半)，且每次运行结果统计时删除了前面 l 个观测数据(最少删除了 1 个，最多删了 96 个)，K 次运行的统计结果是比较理想的，并未出现随 K 值加大总均值 $\bar{\bar{Y}}(K,m,l)$ 偏离理论稳态值的现

象，而置信区间自然越来越小。

10.4 系统性能比较

前面，我们就某一个仿真模型的结果分析进行了讨论。然而仿真研究的大量课题是(特别是在系统设计阶段)需要比较多种不同的方案以便从其中选择最佳方案或可行方案。例如，我们在第 7 章中讨论的库存系统，需要比较九种不同的库存策略。

在进行九种不同的库存策略比较时，我们对每一种策略进行 120 个月的仿真，根据每次仿真运行的结果，我们得到的结论是第 2 种策略(20,60)最好，因为其平均总费用最少。那么这种通过一次仿真所得到的结论是否可信呢?

该库存系统的顾客需求是随机变量，订货延迟时间也是随机变量，从而，每一次仿真运行的结果如平均总费用，平均订货费用，平均保管费用，以及平均短缺损失费用也应是随机变量，那么每种策略仿真运行的结果的差值也应是随机变量。例如平均总费用这一项，在(20,60)这种策略下为 116.84，而在(20,80)这种策略下为 121.45，两者的差值为 4.61，这三个数均是随机变量。显然仅根据这样一个样本来判断是难于保证(20,60)这种策略肯定较(20,80)要好，而应当利用适当的统计分析技术来加以校验。

10.4.1 两系统性能比较

本节我们讨论两系统性能比较技术。两系统性能比较技术的基本思想是建立差值的置信区间。即对每一个系统分别独立地运行 n 次，各自得到同一性能的 n 个样本值，然后建立对应样本差值的置信区间。

设系统 $i(i=1,2)$的 n 个样本为 $x_{i1},x_{i2},\cdots,x_{in},\mu_i=E(x_{ij})(j=1,2,\cdots,n,i=1,2)$为系统的性能期望值，则 $\varepsilon=\mu_1-\mu_2$ 的置信区间可采用如下方法得到：

令 $Z_j=x_{1j}-x_{2j}(j=1,2,\cdots,n)$，则 Z_j 为独立同分布的随机变量，$\varepsilon=E(Z_j)$。由 $\bar{Z}(n)=\sum_{j=1}^{n}Z_j/n, S^2(n)=\frac{1}{n-1}\sum_{j=1}^{n}[Z_j-\bar{Z}(n)]^2$，设置信水平为 α，则 $\hat{\varepsilon}$ 近似 $100(1-\alpha)\%$ 的置信区间为

$$\bar{Z}(n)\pm t_{n-1,1-\alpha/2}\sqrt{S^2(n)/n}$$

如果 Z_j 是正态分布的随机变量，则该置信区间是准确的，即以 $1-\alpha$ 的概率包含 ε，否则根据中心极限定理，当 n 足够大时，该区间包含 ε 的概率趋近 $1-\alpha$。

值得注意的是，我们不必假设 x_{1j} 与 x_{2j} 是独立的，也不必假设 $\mathrm{Var}(x_{1j})$ 与 $\mathrm{Var}(x_{2j})$ 相等；实际上，假若 x_{1j} 与 x_{2j} 是正相关的，则可以减少 $\mathrm{Var}(Z_j)$，从而使置信区间更小。

例 10.8 讨论第 7 章中的库存系统，比较两种库存策略(20,60)，(20,80)，每次仿真运行长度 60 个月，分别运行 10 次，仿真运行的结果如表 10.8 所示。

表 10.8 两种库存策略下的库存系统仿真结果比较

策略(1)	平均费用	策略(2)	平均费用	平均费用差
(20,60)	69.77	(20,80)	70.47	0.70
(20,60)	75.35	(20,80)	74.46	−0.89
(20,60)	68.52	(20,80)	70.20	1.67
(20,60)	69.34	(20,80)	71.20	1.86
(20,60)	63.70	(20,80)	70.60	6.90
(20,60)	69.06	(20,80)	72.41	3.35
(20,60)	72.42	(20,80)	75.64	3.23
(20,60)	73.24	(20,80)	70.28	−2.96
(20,60)	70.07	(20,80)	72.26	2.19
(20,60)	70.41	(20,80)	69.31	−1.10
总平均值	70.19	总平均值	71.68	1.50

■

从上面的仿真结果可以看到,(20,60)这种库存策略的平均每月总费用与(20,80)这种策略的平均每月总费用的差是随机变化的。因而仅仅根据某一次运行的结果来判断哪种策略为好有时难免得出错误的结论。例如在第 1 次运行时两者的差值为 0.70,而在第 2 次运行时的差值则为−0.89。我们只能通过构造置信区间的方法加以比较,才能得到可信的结论。对于本题,记 Z_j 表示两种策略运行后平均每月总费用的差值,下面我们来构造 $\bar{Z}(10)$ 的置信区间。由

$$\bar{Z}(10)=\sum_{j=1}^{10}Z_j/10=1.50$$

$$S_Z^2(10)=\frac{1}{9}\sum_{j=1}^{10}[Z_j-\bar{Z}(10)]^2$$

$$=7.690\,41$$

则当 $\alpha=0.10$ 时,ε 的置信区间为

$$\bar{Z}(10)\pm t_{9,0.95}\sqrt{S_Z^2(10)/10}=1.50\pm1.61$$

即我们有 90%的置信度相信,两种策略的误差在[−0.11,3.11]区间内,这样我们可以说,从总体上看,(20,60)这种策略较(20,80)更好一些。

这种比较两系统性能的方法实质上是将两系统问题简化为单一系统的 Z_j 问题,因而可以采用前面讨论的各种分析方法。

10.4.2 多系统性能比较

实际仿真时可能会产生多种方案(如 K 种,$K>2$)的仿真结果,需要比较这 K 种结果的优劣。多系统择优本质上是参数优化问题。离散事件系统参数优化是一个非常困难的问题,到目前为止仍然未能得到很好地解决,特别是多参数的优化问题,原因在于离散事件系统的随机性。

本节讨论从 K 个方案中选择一个作为某种意义上的最佳系统,并且确知是以一个指定

概率作出的正确选择。这里介绍两种方法，即 Bonferroni 法和“两阶段抽样”法。

1. Bonferroni 法

设有 K 个系统方案对于某一规定的性能参数 $E(Y_i)$，$i=1,2,\cdots,k$ 进行比较，如果采取某一方案 j 作为比较的基础，可以建立 $c=k-1$ 个 $E(Y_i)-E(Y_j)$ 的置信区间，其置信度为 $1-\alpha_i$，$i=1,2,\cdots,c$。

令 S_i 为一个声明事件，即 $S_i=$｛给出的置信区间中包含所仿真的参数｝，则 $P\{S_i \text{ 为真}\}=1-\alpha_i$，我们希望在作多方案比较时，所有方案有关参数比较的声明为真的概率较高，在这样的条件下进行比较，可使比较结果有较高的置信度。为此建立以下准则，称为 Bonferroni 不等式准则

$$P\{\text{所有 } S_i \text{ 同时为真}, i=1,2,\cdots,c\} \geqslant 1-\sum_{i=1}^{c}\alpha_i = 1-\alpha_E \tag{10.30}$$

式中 $\alpha_E=\sum_{i=1}^{c}\alpha_i$ 称为总误差概率，上式也可写成

$$P\{\text{一个或多个置信区间不包含所预计的参数}\} < \alpha_E \tag{10.31}$$

因此，α_E 是作出错误结论的概率上限。

当进行 c 次方案比较时，首先应按实际问题的要求确定 α_E，按照 Bonferroni 不等式准则，如果所有 α_i 均选为相等时，则 $\alpha_i=\alpha_E/c$，如果 $\alpha_i(i=1,2,\cdots,c)$ 之间不相等，则必须满足 $\sum_{i=1}^{c}\alpha_i=\alpha_E$。

显然，在采用 Bonferroni 比较准则时，要求每次比较的 α_i 准则都要小于实际问题的 α_E。例如，当需要做 10 个方案比较时，若实际问题要求的置信度为 0.95，则 $\alpha_i=0.05/10=0.005$，故每次比较的置信度将为 0.995，这显然会使置信区间变宽。或者说，要达到一定的精度，需要做较多次的仿真运行。为避免这一情况，通常 $c<10$，而在实际应用中 $c=2\sim3$ 就足够了。

2. “两阶段抽样”法

同两系统比较一样，令 X_{ij} 为第 i 个系统的第 j 次实验所获得的随机变量，并令 $\mu_i=E(X_{ij})$，假设 $X_{i1}, X_{i2},\cdots$ 都是独立同分布随机变量，并且不同系统的运行必须是相互独立进行的，i 的取值范围可以从 1 到 K，不失一般性，假定 $\mu_{i1}\leqslant\mu_{i2}\leqslant\cdots\leqslant\mu_{ik}$，在本节我们的目标就是选择一个具有最小期望响应 μ_{i1} 的系统。令 CS 表示“正确选择”这一事件。

观察值 X_{ij} 的固有随机性就意味着我们不可能绝对肯定我们做出了 CS，但我们希望能预估 CS 的概率，进而，如果 μ_{i1} 与 μ_{i2} 非常接近，那么我们是否错误地选择了系统 i_2 就变得无关紧要，因此我们也需要有一种方法来避免为区别这种不重要的区别而做大量实验。综合起来，我们所要研究的问题的确切描述是：假如 $\mu_{i2}-\mu_{i1}\geqslant d^*$，我们希望 $P\{CS\}\geqslant P^*$，其中最小 CS 概率 $P^*>1/k$ 以及“无关紧要量”$d^*>0$，都由分析人员确定。

解决这一问题的方法是对 K 个系统中的每一个进行两步采样。第一步我们对每个系统规定一个固定的实验次数。然后利用方差估计的结果确定在第二步每个系统需要增加多少次重复运行。我们假定 X_{ij} 符合正态分布，在第一阶段采样中，我们对 K 个系统中的每一

个做 n_0 次实验，并且定义第一阶段样本的均值和方差分别为

$$\overline{X}_i^{(1)}(n_0)=\frac{\sum_{j=1}^{n_0}X_{ij}}{n_0} \tag{10.32}$$

$$S_i^2(n_0)=\frac{\sum[X_{ij}-\overline{X}_i^{(1)}(n_0)]^2}{n_0-1}\qquad(i=1,2,\cdots,k) \tag{10.33}$$

则我们计算系统 i 所需的总样本容量为 N_i 为

$$N_i=\max\left\{n_0+1,\left\lfloor\frac{h_1^2S_i^2(n_0)}{(d^*)^2}\right\rfloor\right\} \tag{10.34}$$

其中，$\lfloor z\rfloor$ 是大于或等于 z 的最小整数，h_1 是取决于 K，P^* 和 n_0 的常数，可以由表 10.9 获得。

表 10.9 “两阶段抽样”法参数表

P^*	n_0	$K=2$	$K=3$	$K=4$	$K=5$	$K=6$	$K=7$	$K=8$
0.90	20	1.896	2.342	2.583	2.747	2.870	2.969	3.051
0.90	40	1.852	2.283	2.514	2.669	2.785	2.878	2.954
0.95	20	2.453	2.872	3.101	3.258	3.377	3.472	3.551
0.95	40	2.386	2.786	3.003	3.150	3.260	3.349	3.422

下一步，我们对系统 $i(i=1,2,\cdots,K)$ 需多做 N_i-n_0 次重复运行，并得到第二阶段的样本均值

$$\overline{X}_i^{(2)}(N_i-n_0)=\frac{\sum_{j=n_0+1}^{N_i}X_{ij}}{N_i-n_0} \tag{10.35}$$

则定义权为

$$W_{i1}=\frac{n_0}{N_i}\left(1+\left(1-\frac{N_i}{n_0}\left(1-\frac{(N_i-n_0)(d^*)^2}{h_1^2S_i^2(n_0)}\right)\right)^{1/2}\right)\qquad(i=1,2,\cdots,k) \tag{10.36}$$

及

$$W_{i2}=1-W_{i1} \tag{10.37}$$

然后定义加权样本均值

$$\widetilde{X}_i(N_i)=W_{i1}\overline{X}_i^{(1)}(n_0)+W_{i2}\overline{X}_i^{(2)}(N_i-n_0) \tag{10.38}$$

这样，具有最小$\widetilde{X}_i(N_i)$的系统就是我们寻找的最佳系统。

10.5 仿真结果分析应用举例

通过前面几节的讨论，我们对离散事件系统仿真结果分析的有关内容有了比较详细的了解，本节通过一个实例来加深这方面的认识。

例 10.9 一加工车间，有 1 台加工中心及 1 台测量机，待加工零件到达加工中心的时间间隔服从指数分布，均值为 1min，每个零件的加工时间服从[0.7,0.8]min 之间均匀分布，测量检验每个零件的时间也服从[0.8,0.9]min 之间的均匀分布，经检验，90%的加工

过的零件通过检验，10%的零件必须返回重新加工。加工中心可能会发生故障而停工，停工时间为均值是 6 小时的指数分布，维修时间为[8,12]min 区间内的均匀分布。系统的初始状态为队长为 0，设备空闲，对该系统进行终止型仿真，仿真时间 16 小时，估计系统的加工能力，每个零件在系统中的时间，同时希望比较当每个零件的加工时间为[0.5,0.6]min 与[0.7,0.8]min 区间上的均匀分布时，系统加工能力及每个零件在系统中的时间。

由于本题的重点是仿真结果分析，因此关于仿真模型的建立及仿真程序不再赘述，下面只给出该系统的示意图(参见图 10.4)。

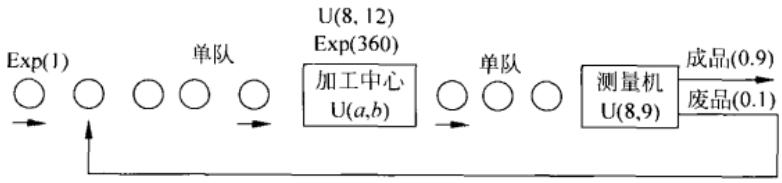


图 10.4 加工车间示意图[(a,b)为(0.7,0.8)或(0.5,0.6)]

在实际中，人们常常用确定性模型代替随机模型以估计系统性能。为了说明这种做法的危险性，我们这里给出确定性模型的仿真结果。该确定性模型是

零件到达时间间隔	1min
零件加工时间(每件)	0.75min
零件测量时间(每件)	0.85min
机床间隔时间	360min
机床返修时间	10min

都是取原随机变量时间的均值，并认为所有零件都是合格品，对此模型进行仿真，得到：

(1) 16 小时内加工零件数 $m=958$；

(2) 每个零件在系统中的等待时间 $T=2.3$min。

考虑到系统中的随机性，我们进行一次仿真长度为 16 小时的终止型仿真，设每个零件在系统中的时间为 T_i，则

$$T(m) = \sum_{i=1}^{m} T_i/m$$

仿真结果为

$$\overline{T}(m) = 10.2\text{min}$$

$$m = 971$$

比较上述两组数据，显然 m 相差较少，但 $\overline{T}(m)$ 则相差很大，那么到底应该相信哪一组数据呢？人们往往先计算第二组数据的方差，根据其方差大小再进行判断。下面我们讨论 $\overline{T}(m)$ 的方差 $\hat{\sigma}(T)$

$$\hat{\sigma}(T) = \sqrt{\frac{1}{m(m-1)}\sum_{i=1}^{m}(T_i - \overline{T}(m))^2}$$

其中，$\hat{\sigma}(T)$是 $\sigma(T)$的估计值。

仿真结果表明 $\sigma(T)=0.23$，显然是相当小的，那么我们能否认为第二组可信呢？通过前面的讨论可以相信，这种结论不一定是正确的。按统计学观点，上述计算方差的公式要求

所使用的数据是独立的，即要求 T_i 是独立的，而在我们讨论的系统中 T_i 则是相关的。直观地看，如果 T_i 较大，则第 $i+1$ 个零件的等待时间很可能也比较长。这样得到的方差估计 $\hat{\sigma}(T)$ 可能是有偏的。解决的方法就是我们在 10.2.1 节中的固定样本长度法。对该模型独立地运行 5 次的仿真结果如表 10.10 所示。

表 10.10 模型独立运行 5 次的仿真结果

n	加工零件数	零件在系统中平均时间/min
1	971	10.2
2	913	8.1
3	996	12.2
4	956	6.9
5	937	9.1
总平均	954.6	9.3

从上述结果可以看到

$$\overline{X}(n) = E(\overline{T}_j(m_j)) = 9.3\text{min} \qquad (j = 1,2,\cdots,5)$$

由于是独立运行，可以得到 $\overline{X}(n)$ 的标准方差的无偏估计为

$$S(n) = \sqrt{\frac{1}{n(n-1)}\sum_{j=1}^{n}[\overline{T}_j(m_j) - \overline{X}(n)]^2}$$

设 $\alpha=0.05$，则得到零件在系统中的时间的置信区间为

$$\overline{X}(n) \pm t_{n-1,0.975}S(n)$$

可得到置信区间为[9.3±2.5]或[6.8,11.8]。那么我们可以有 95%的置信度认为零件在系统中的时间为[6.8,11.8]区间上。而采用确定性模型得到的数据 T 是不可信的。同样我们也可建立加工零件数目的置信区间为[915.2,994.0]，采用确定性模型对 m 计算得到的数据在该区间内，因而可以认为是可信的。

上面是终止型仿真，如果我们希望得到系统的稳态性能而不希望进行稳态型仿真的话，我们可以采用重复删除法得到零件在系统中的时间如下

$\overline{T}_1(m_1) = 12.8$，$\overline{T}_2(m_2) = 9.5$，$\overline{T}_3(m_3) = 11.1$，$\overline{T}_4(m_4) = 7.0$，$\overline{T}_5(m_5) = 8.8$

得到 $\overline{X}(5)=9.8$，在 $\alpha=0.05$ 下置信区间为

$$\overline{X}(5) \pm t_{4,0.975}\sqrt{\hat{\sigma}^2(5)} = 9.8 \pm 2.7$$

从而我们可以有 95%的置信度认为，该系统在稳态下，零件在系统中的时间在[7.1,12.5]区间上。

下面我们讨论当加工中心加工零件的时间从[0.7,0.8]的均匀分布加快到[0.5,0.6]的均匀分布时，两个系统性能的比较。

从 10.4 节中讨论我们知道，我们应在相同的仿真条件下进行比较，即对新的系统也独立地进行 5 次终止型仿真，其仿真结果如表 10.11 所示。

用 $D_{1j}(j=1,2,\cdots,5)$ 表示两个系统加工零件数之差，$D_{2j}(j=1,2,\cdots,5)$ 表示零件在系统中平均时间之差，则得到表 10.12。

表 10.11 独立进行 5 次终止型仿真的结果

n	加工零件数	零件在系统中平均时间/min
1	942	6.6
2	956	4.8
3	965	13.3
4	955	6.2
5	943	9.9
总平均	952.2	8.16

表 10.12 两个系统加工零件数之差与零件在系统中平均时间之差

n	加工零件数之差 D_{1j}	零件在系统中平均时间之差 D_{2j}/min
1	29	3.6
2	−43	3.3
3	31	−1.1
4	1	0.7
5	−6	−0.8
总平均	2.4	1.14

可分别得到两系统差别的置信区间为

$$D_1 = 2.4 \pm t_{4,0.975}\sqrt{\sum_{j=1}^{5}(2.4 - D_{1j})^2/20} = 2.4 \pm 37.55$$

$$D_2 = 1.14 \pm t_{4,0.975}\sqrt{\sum_{j=1}^{5}(1.14 - D_{2j})^2/20} = 1.14 \pm 2.754$$

由上述结果可以看到，在统计意义下，略微提高加工中心的加工速度对加工零件的数目及缩短零件在系统中的时间并没有明显的效果。而如果仅仅依据两个系统的第一次运行结果进行比较则会得出不正确的结论。 ■

10.6 方差减少技术*

由于仿真模型输入变量(如到达时间间隔、服务时间、转运时间等)大多都是随机变量，受其影响，仿真模型的输出变量(比如等候时间、运转周期等)也不可避免地成为随机变量，因此仿真得到的输出变量数值也呈现随机变化的行为。这种变化对输出变量的估计也产生影响，比如必须用统计估算的方法来估计这些输出变量的均值和方差，并用这些统计量来表征输出变量的实际变化。当这些统计量的误差变化很大时，就会影响到它们的置信度或者说可靠程度，从而影响到它们的代表性及应用。因此，在分析仿真输出结果时，人们往往希望能够减少由于输出指标数值的随机波动而带来的变化误差，从而提高统计估计的精度与置信度。

有两种基本方法可以用来达到这个目的：一个就是增加仿真运行的次数(也即增加仿真实验的样本容量)，这个方法往往需要增加仿真试验的时间和成本；而另一个有效的办法

就是采用引入相关性的误差变化减少方法，当仿真的系统大而复杂时，这一方法往往更为经济有效。

在对复杂系统进行仿真时，获得精确的估计是系统仿真的重要问题之一，而估计的精度常常用一组输出样本的方差进行评价。为了改进精度，有必要在仿真中采用方差减少技术。常用的方差减少技术有：公共随机数法、对偶变量法、控制变量法，其他还有间接估计法和条件期望法。本书介绍前三种方法。

10.6.1 公共随机数法

公共随机数法(common random numbers，CRN)主要用于比较两个不同系统模型的差异。对于不同的系统模型来说，所得到的仿真输出结果是有差异的。引起这种差异的主要原因是系统模型结构上的差异以及随机因素的影响。公共随机数法通过采用相同的随机输入，尽可能消除由于随机因素造成的仿真输出结果的差异，而使这种差异主要表现为系统模型本身的差异。

采用公共随机数法的基本思想是这样的：当比较不同的方案时，如果比较是在“相同或者类似的”试验条件下进行的，人们就更容易相信观察到的结果(不同之处)是由于方案本身的特征不同造成的，而不是由于比较时试验条件的差异或者输出数值随机变化的波动造成的，或者说比较结果的可信度就高一些。例如在图 10.5 中可以看到，尽管两种方案的输出数据都呈现随机波动，但由于方案本身的特征不同而造成的差异(系统特征差异)是非常明显的(其中 μ_1 和 μ_2 是两种方案各自的期望值)。但如果由于试验条件差异而造成的数值随机波动太大，那么真正的系统特征差异就会很难辨别或准确地估计。在仿真过程中，造成方案比较时试验条件的差异和波动就是随机数值的产生及其应用。事实上，在整个仿真时间内，一直在不断地产生和使用随机数与随机数值，比如在仿真一个库存管理系统时，需要不停地产生需求随机到达的时间以及随机变化的需求数量。如果能够使仿真不同方案时随机数值(例如对应于同样的输出变量)的产生和应用尽量一致，那么就可以减少由于仿真试验条件(产生和应用随机数值)的差异而造成的不同，从而使比较结果的精度提高。

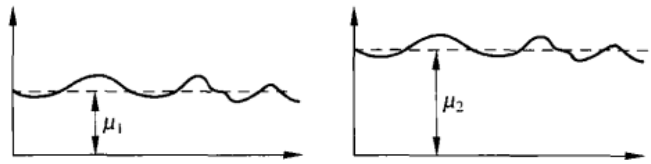


图 10.5 方案特征的不同造成的差异(左图)和数值随机波动(右图)造成的差异

设两个系统模型中表征某个系统性能测度的输出随机变量分别为 Y_1，Y_2，对两个系统模型分别进行 n 次独立重复仿真运行，用 Y_{1i}，Y_{2i} 表示第一个和第二个系统模型在第 i 次独立重复运行中得到的观察值($i=1,2,\cdots,n$)，希望用这些数据得到 $\mu=E(Y_1)-E(Y_2)$ 的估计。

令 $Z_i=Y_{1i}-Y_{2i}(i=1,2,\cdots,n)$，则 $E(Z_i)=E(Y_1)-E(Y_2)$，所以，$\bar{Z}_n=\frac{1}{n}\sum_{i=1}^{n}Z_i$ 是 μ 的无偏估计。由于 $Z_1,Z_2,\cdots,Z_n$ 是独立同分布的随机变量，因此有

$$\mathrm{Var}(\bar{Z}_n)=\frac{1}{n}[\mathrm{Var}(Y_{1i})+\mathrm{Var}(Y_{2i})-2\mathrm{Cov}(Y_{1i},Y_{2i})] \tag{10.39}$$

在仿真运行中，对两个系统模型分别采用独立的随机输入，则输出 Y_1 与 Y_2 也相互独立，从而 $\mathrm{Cov}(Y_{1i},Y_{2i})=0$。由式(10.39)易见，若 $\mathrm{Cov}(Y_{1i},Y_{2i})>0$，即 Y_{1i},Y_{2i} 正相关，则可使方差 $\mathrm{Var}(\bar{Z}_n)$ 减小。所以从理论上来说，只要设法使两个系统模型的输出 Y_{1i},Y_{2i} 正相关，就能达到减少方差的目的。

公共随机数法在不同系统模型的仿真试验中，采用同步的随机数，即不同系统模型中的相同特征的输入随机变量用同一发生器，同一个种子值生成。例如比较两个排队系统的性能差异，通常采用同步随机数的方式有三种：

① 只对到达间隔时间变量用同一随机数流生成；

② 只对服务时间变量用同一随机数流生成；

③ 到达过程和服务过程都实现同步的公共随机数仿真。

公共随机数法实现方差减少的效果主要取决于模型本身的特征。由于受到具体模型的限制，还不能说公共随机数法一定能达到方差减少。

为了实施公用随机数法，需要使各个模型中的随机数同步，即在一个模型中使用于一个具体目的的随机数，在所有其他模型中也应该使用于同一目的，在仿真中达到这种同步的一般原则如下：

(1) 如果能够有几个可以同时工作的不同随机数发生器，则可以用一个发生器专门为一个指定的随机变量产生种子。不同的随机变量用不同的随机数发生器。

(2) 实现产生出所需要的随机数并存储起来，在对各个模型仿真运行时按照需要取用这些随机数。

(3) 使用逆变换法产生随机变量，因为这种方法每产生一个随机变量仅仅只需要一个单位均匀分布的随机数。在系统仿真中要采用公共随机数法作不同方案的仿真运行，主要取决于模型的结构和参数以及产生随机数的方法。当模型在结构上有所不同但输入随机变量相同时，可在仿真程序中同时调用若干个不同的随机数发生器，不同参数用不同的随机数发生器，但在不同方案的仿真运行中，对应参数都调用同一随机数发生器，并保持其同步运行。例如，不同方案的到达间隔时间都从同一起点，调用同一个随机数发生器。类似地服务时间也调用另一同步随机数发生器等。这种方法称为完全同步的公共随机数法。当模型在结构上差异较大，往往不易实现完全同步的公共随机数仿真，这时也可采取部分参数同步的公共随机数仿真，这也有助于缩减方差。

那么 CRN 究竟是如何在仿真不同的方案过程中引入正的相关性呢？具体的做法很简单：CRN 在随机数的产生和使用上遵循一条“同步原则”，即在每一次仿真运行中，对参与比较的所有方案，同步地使用随机数。比如，如果在某一方案中用一段随机数$[U_j,\cdots,U_m]$来产生仿真加工服务时间，那么在所有其他的方案中，也采用同一段随机数来产生和仿真加工服务时间。这种做法也被称为“随机数段专用法”(stream dedication)：把仿真系统产生的随机数列分成若干段，然后把每一段专用于某一个或某一类随机变量的应用(比如产生某类个体的到达时间间隔)。表 10.13 显示了这个基本思想。

表 10.13 随机数段专用的例子

随机数段 1	随机数段 2	随机数段 3
$U_1,\cdots,U_{100\,000}$	$U_{100\,001},\cdots,U_{200\,000}$	$U_{200\,001},\cdots,U_{300\,000}$
到达时间间隔	服务加工时间	转运时间

有些离散事件仿真软件的随机数序列的周期很长，可以分段使用。例如，Arena 软件按现有的随机数发生器产生的随机数序列的周期是 3.1×10^{57}；这个庞大的周期被分为若干个“很长”的分段(分段长度达 1.7×10^{38})。可以在设计输入变量的时候根据随机数专用法来分配随机数的使用，比如将分段 3 指定用于产生某类个体的到达时间间隔等。对于任何输入变量的分布产生，如果指定专用的随机数段，则系统将自动把分段 10 用于该目的。

例 10.10 系统是一个由 5 个加工单元，每个加工单元有数台机床组成的制造车间。分析要求对于给定的零件到达规律(数量、工艺路线和时间)，比较 3 种不同的设备添置方案。比较的指标是加工零件在系统平稳运行的条件下总的平均等候时间(或延误时间)。如果以 T_j 代表在第 j 个方案下运行的平均延误时间，则需要估计的是下面的统计差别

$$D(1,2)=T_1-T_2 \quad D(1,3)=T_1-T_3 \quad D(2,3)=T_2-T_3$$

对每一个方案，仿真重复 10 次，每次的运行长度是 $100d$，其中的前 $10d$ 考虑为热身区间而被截去，所以用于输出分析的数据是基于 $10d$ 以后的仿真结果。研究者采用了独立抽样和 CRN 两种方法，并且就统计估算的误差变化对比了分析结果(见表 10.14)(其中 $S^2(i,j)$ 是方差 $\mathrm{Var}(D(i,j))$的无偏估计量)。 ■

表 10.14 比较独立抽样和 CRN 分析的误差变化

样本的方差	独立抽样	CRN	方差减少(%)
$S^2(1,2)$	6.27	1.46	77
$S^2(1,3)$	10.40	2.35	77
$S^2(2,3)$	23.08	0.87	96

从表 10.14 中可以看出，应用 CRN 的分析显著地减少了统计量的样本方差，达到 77% 之多，从而有效地提高了统计分析的精度。如果要用独立抽样的方法来减少样本方差以达到同样的水平，比如说 $S^2(1,2)$，则需要将仿真的次数提高 4.29(=6.27/1.46)倍之多！可见，采用 CRN 显著地改善了统计分析的效率。

例 10.11 某服务系统可以采用一台效率比较高的设备，构成 M/M/1 排队系统，也可采用两台效率较低的设备，构成 M/M/2 单队列排队系统。若不同方案的系统的服务率相同，均为 $\rho=0.9$，系统的性能测度为前 100 名顾客的平均等待时间。

为了比较不同仿真运行的输出参数的方差，分别对两个方案作不同的仿真运行。

(1) 用独立随机数流对两个方案分别作 100 次独立重复运行，记为方案(I)。

(2) 只对到达间隔时间实现部分参数同步的公共随机数仿真，其他参数仍保持独立随机数仿真，记为方案(A)。

(3) 只对服务时间实现参数同步的公共随机数仿真，其他参数仍保持独立随机数仿真，记为方案(S)。

(4) 到达过程和服务过程都实现同步的公共随机数仿真，记为方案(A,S)。对两个方

案分别用以上四种仿真方式作 100 次重复运行，相应的输出观察值为

$$z_j = y_{1j} - y_{2j}, \quad j = 1,2,\cdots,100$$

z_j 的方差为

$$S^2 = \left\{ \frac{\sum_{j=1}^{100} [z_j - \bar{z}(100)]^2}{(100-1)} \right\}$$

其数值如表 10.15 所示。

表 10.15 四种仿真方式作 100 次重复运行的结果

仿真方式	I	A	S	(A,S)
样本方差 S^2	25.491	11.647	10.473	0.103

■

可见，在完全同步时，其输出参数的方差将比独立运行时要降低 99.6%。而在达到完全同步运行的仿真精度时，独立仿真运行需要增加 247 次重复仿真运行。由此可见采用公共随机数法对提高仿真效率的效果。

但是，公共随机数法对缩减方差的幅度主要取决于模型本身的特征，目前还不能给出关于公共随机数法对缩减方差的一般表达式，此问题有待于进一步研究。

10.6.2 对偶变量法

对偶变量法(antithetic variates method，AVM)是用于单一系统仿真的方差减少技术。采用对偶变量法的目的是尽可能消除随机因素对仿真输出结果产生的影响。这种方法对系统模型进行成对的仿真运行，使第一次运行中较小的观察值能抵消第二次运行中较大的观察值，或用较大的观察值抵消第二次运行中较小的观察值。总之，将一对观察值的平均数作为估计的基本数据，希望能缩小估计量的方差。

AVM 的基本思想是：假设在第一次仿真运行时采用了随机数$[U_1,U_2,\cdots,U_n]$，那么在第二次(重复)运行时，则采用随机数$[1-U_1,1-U_2,\cdots,1-U_n]$，如果 U 是一个随机变量，那么显然 $1-U$ 也是一个随机变量，而且如果 U 值越大，则相应的 $1-U$ 值就越小，反之亦然，所以由此产生的随机变量以及仿真的结果将会是呈负相关的，即在第一次和第二次的仿真运行之间产生了呈负相关的输出结果。以此类推，在第 3 次仿真时采用独立的随机数，而在第 4 次则使用第 3 次使用的随机数的对偶数，等等。这样每一对搭配的两次运行便可以产生一个更接近于真值的平均值，减少了误差的变化，从而使最后的总体统计估算精度得到提高。同 CRN 一样，对偶变量法也要求同步地使用随机数(比如应用随机数分段专用的办法)。

设对系统模型共作 n 对独立重复仿真运行，所得仿真输出随机变量 Y 的观察数据为 $(Y_1^{(1)},Y_1^{(2)}),(Y_2^{(1)},Y_2^{(2)}),\cdots,(Y_n^{(1)},Y_n^{(2)})$，其中 $Y_1^{(1)},Y_2^{(1)},\cdots,Y_n^{(1)}$ 相互独立，$Y_1^{(2)},Y_2^{(2)},\cdots,Y_n^{(2)}$ 也相互独立，共作 $2n$ 次仿真运行。

令

$$Y_i = \frac{Y_i^{(1)} + Y_i^{(2)}}{2}, (i = 1,2,\cdots,n) \tag{10.40}$$

易见，$\bar{Y}_n=\frac{1}{n}\sum_{i=1}^{n}Y_i$ 是 $\mu=E(Y)$ 的无偏估计。由于 $Y_1,Y_2,\cdots,Y_n$ 相互独立，故有

$$\mathrm{Var}(\bar{Y}_n)=\frac{1}{n}[\mathrm{Var}(Y_i^{(1)})+\mathrm{Var}(Y_i^{(2)})+2\mathrm{Cov}(Y_i^{(1)},Y_i^{(2)})] \tag{10.41}$$

若在成对运行中分别采用独立的随机输入，则输出 $Y_i^{(1)}$ 与 $Y_i^{(2)}$ 也独立，此时 $\mathrm{Cov}(Y_{1i},Y_{2i})=0$。由式可以看出，只要能设法使 $Y_i^{(1)}$ 与 $Y_i^{(2)}$ 负相关，即 $\mathrm{Cov}(Y_{1i},Y_{2i})<0$，就能实现方差减少。

较之公共随机数法，对偶变量法实现方差减少的效果更取决于系统模型。通常，对偶变量法成功与否与估计量所依赖的函数特性有很大关系。如果已知函数性质，则可相应地构造对偶变换，这样既保证了估计的可靠性，又大大提高了精度。

10.6.3 控制变量法

控制变量法(control variates method,CVM)是利用随机变量之间的相关性来实现方差减少的目的。

设 Y 是某一输出随机变量，我们要估计 $\mu=E(Y)$，而 X 是一个均值 $\nu=E(X)$ 已知，且与 Y 相关的随机变量。令

$$Y_c=Y-a(X-\nu) \tag{10.42}$$

其中，a 为任意实数，易见 Y_c 也是 μ 的无偏估计。由于

$$\mathrm{Var}(Y_c)=E[Y-a(X-\nu)]^2=\mathrm{Var}(Y)+a^2\mathrm{Var}(X)-2a\mathrm{Cov}(X,Y) \tag{10.43}$$

若 $a^2\mathrm{Var}(X)-2a\mathrm{Cov}(X,Y)<0$，则 Y_c 的方差将小于 Y 的方差，称 X 为 Y 的控制变量，因为 X 部分地控制了 Y。

从上面的讨论可以看出，控制变量法的两个关键问题是寻找适当的控制变量 X 和确定系数 a，使 $\mathrm{Var}(Y_c)$ 尽可能小。

优良的控制变量 X 应当与 Y 是强相关的，这取决于模型本身的结构。例如，在排队系统中取输出变量 Y 为顾客的平均等待时间，则控制变量可以选用已知分布的输入变量。当选用到达间隔时间为控制变量，则它与 Y 之间构成负相关关系。若选用服务时间为控制变量，则它与 Y 之间构成正相关关系。利用这种控制变量的优点是它们在仿真中必须产生，故不必另花时间去专门产生这种随机变量。

为确定系数 a，在式(10.43)中对 a 求导，得

$$\frac{\mathrm{dVar}(Y_c)}{\mathrm{d}a}=2a\mathrm{Var}(X)-2\mathrm{Cov}(X,Y)$$

令上式为零，得

$$a^*=\frac{\mathrm{Cov}(X,Y)}{\mathrm{Var}(X)}$$

将 a^* 代入式(10.43)得

$$\begin{aligned}\mathrm{Var}(Y_c)&=\mathrm{Var}(Y)+\left[\frac{\mathrm{Cov}(X,Y)}{\mathrm{Var}(X)}\right]^2\times\mathrm{Var}(X)-2\left[\frac{\mathrm{Cov}(X,Y)}{\mathrm{Var}(X)}\right]\times\mathrm{Cov}(X,Y)\\&=\mathrm{Var}(Y)-\frac{[\mathrm{Cov}(X,Y)]^2}{\mathrm{Var}(X)}\end{aligned} \tag{10.44}$$

当 X 与 Y 相关时，$\mathrm{Cov}(X,Y)\neq 0$，从而由式(10.44) 易见 $\mathrm{Var}(Y_c)<\mathrm{Var}(Y)$，即只要能找到与 Y 相关的控制变量 X，就能实现方差减少。从式(10.44)还可以看出，控制变量 X 与 Y 的相关性越强，方差减少的效果越好。

10.7 小结

离散事件系统仿真是统计试验，因此，只有通过恰当地进行统计分析，才能使得我们通过离散事件系统仿真产生的数据得出合理的结论。仿真试验的目的是得到所研究系统的性能，统计分析的目的是为离散事件系统仿真分析提供保障。

本章讨论了终止型仿真和稳态型仿真的区别。然后分别讨论单个系统的终止型仿真与稳态型仿真的分析方法，介绍了多个系统仿真结果的比较技术，为了加深读者对仿真结果分析的理解，我们通过一个实例综合运用了本章的几种仿真输出分析方法，最后，向读者介绍了提高仿真分析效能和精度的方差减少技术。

习题

1. 为什么说终止型仿真时，即使独立运行次数相当多仍难以得到系统的稳态性能？

2. 试列表比较终止型和稳态型仿真应用条件、结果分析方法，包括运行次数、基本算法。

3. 表 10.16 是某一 M/G/1 排队系统仿真运行后所得到的队列中人数仿真结果。共运行了 10 次，每次运行长度为 15 000min，按批统计，每批长度为 1 000min。试对每次运行结果分别用批均值法计算其平均队长及其方差。然后用重复删除法，删除的批数依次分别为前 1,3,2,5,6,4,11,7,9,12，再计算其平均队长及其方差，最后比较两种方法的结果。

表 10.16 某一 M/G/1 排队系统仿真结果

运行次数	子区间														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	3.61	3.21	2.18	6.92	2.82	1.59	3.55	5.60	3.04	2.57	1.41	3.07	4.03	2.70	2.71
2	2.91	9.00	16.1	24.5	25.2	21.6	24.5	8.45	8.53	14.8	23.6	27.6	24.2	8.58	4.06
3	7.07	19.5	20.6	8.11	12.6	22.1	14.1	9.87	24.0	24.5	14.6	6.08	4.82	16.0	23.4
4	6.62	1.75	12.9	8.77	1.25	1.16	1.92	6.29	4.74	17.4	18.2	18.6	4.62	2.76	1.57
5	2.16	1.32	2.14	2.18	2.59	1.20	4.11	6.21	7.31	1.58	2.16	3.08	2.32	2.21	3.32
6	0.93	3.54	4.80	0.72	2.95	5.56	1.96	2.07	2.74	3.45	14.2	13.4	7.87	0.94	3.19
7	1.12	2.59	5.05	1.16	2.72	5.12	5.03	4.14	4.98	15.8	9.29	2.14	8.72	29.8	28.9
8	1.54	5.94	5.33	2.91	2.69	1.91	3.27	3.61	10.4	9.66	4.13	6.14	7.90	2.61	7.95
9	8.93	4.78	0.74	2.56	9.43	18.6	8.14	1.49	4.51	1.69	12.7	11.3	3.32	3.42	3.35
10	4.78	3.84	10.4	5.87	1.01	2.59	16.8	27.3	26.8	20.9	7.26	2.32	5.04	3.50	9.11

4. 与传统的独立抽样方法比较，CRN 的优点何在？CRN 方法和 AVM 方法的主要区别是什么？如果需要比较几个不同的方案，应该采取哪一种方法？为什么？

5. 考虑单服务台排队系统，其顾客到达服从速率为每分钟 $\lambda=10.82$ 的泊松分布，服务时间服从均值 5.1s，方差 $0.98s^2$ 的正态分布。当一个顾客到达时，他发现系统中先到的顾客还有 i 个，估计出该顾客在系统中的平均时间，即估计

$$w_i = E(W \mid N = i) \quad i = 0,1,2,\cdots$$

其中 W 是典型系统时间，N 是到达时发现的顾客数。例如，w_0 是到达时发现系统为空的那些顾客的平均系统时间，w_1 是到达时发现系统中有 1 个顾客的那些顾客的平均系统时间，依此类推。w_i 的估计 $\hat{w}_i$ 将是所有到达时发现系统中有 i 个顾客的那些顾客的系统时间样本均值。画出 $\hat{w}_i$ 相对于 i 的图，假设并检验 w_i 和 i 之间的关系。

① 在空闲和空的初始状态下，仿真周期 10 小时。

② 在 1 小时初始阶段之后，再仿真周期 10 小时。①和②的结果有明显的不同吗？

③ 设服务时间服从均值为 5.1s 的指数分布，重复①和②。

④ 设服务时间固定为 5.1s，重复①和②。

在估计 $w_0, w_1, \cdots, w_6$ 时，要使得每个标准误差至多为 3s，那么至少需要进行多少次重复运行。利用这个运行次数，重复①到④。

6. 考虑下面的库存系统

① 一旦库存水平等于或者低于 10 件，就发出一个订货单，在同一时间，只能有一个订货单未完成。

② 每个订单的订货量为 Q，在仓库中每件货物每天的维护费用为 0.50 美元，每次订货的费用固定为 10.00 美元。

③ 提前期服从 0 到 5 天之间的离散均匀分布。

④ 如果一个需求出现在库存为 0 的时期，则销售损失为每件 2.00 美元。

⑤ 每天的顾客数量服从以下分布

每天的顾客数量	概率	每天的顾客数量	概率
1	0.23	3	0.22
2	0.41	4	0.14

⑥ 每个顾客的需求服从均值为 3 件的泊松分布。

⑦ 为了简化问题，假设所有的需求都发生在中午，随后所有的订单都立即发出。

进一步假设在下午五点或者当天需求发生之后接收到订货。采用的策略为 $Q=20$。进行 5 次重复独立运行，每次运行长度为 100 天，计算长运行日平均费用的 90%的置信区间，研究初始库存水平和未完成订单的存在对估计日平均费用的影响。在初始库存水平为 $Q+10$ 且没有未完成订单的条件下开始仿真。

参考文献

[1] Law A M, Kelton W D. Simulation Modeling and Analysis. McGraw Hill, 1982

[2] 鲍居武. 仿真数值分析方法及应用. 北京：学苑出版社，1993

[3] 冯允成,邹志红,周泓. 离散系统仿真. 北京: 机械工业出版社,1998

[4] Fishman G S. Grouping Observation in Digital Simulation. Management Science, 1978, 24

[5] Kabak I W. Stopping Rules for Queuing Simulation. Operation Research, 1968, 16

[6] Law A M. Confidence Intervals for Steady-State Simulations: L A Survey of Fixed Sample Size Procedures. Technical Report,78-5,University of Wisconsin,1978

[7] Kelton W D. Statistical Analysis Methods Enhance Usefulness Reliability of Simulation Models. Industrial Engineering, 1986, 9

[8] Banks J,Carson J S 著. 肖田元,范文慧译. 离散事件系统仿真. 北京: 机械工业出版社,2007

第三篇　现代仿真篇

第 11 章
并行仿真与分布仿真*

11.1 前言

随着仿真研究对象日益复杂，规模日益增大，尽管计算机技术在突飞猛进地发展，但仍然满足不了应用的要求。人们分析过，单一 CPU 的数字计算机的计算速度不可能超越电路信号传输速度的极限，即电场传播速度 3×10^8 m/s。如果传输线按 1m 计算，则计算机的反应速度不可能超过每秒 3×10^8 次操作。然而，三自由度空间飞行器的实时仿真要求大于每秒 10^7 次操作，一个核反应堆的仿真要求大于每秒 10^{10} 次操作，而大系统的优化研究则要求大于每秒 10^{12} 次操作，等等。

显然，单一 CPU 的计算机越来越难以满足仿真的要求。仿真技术的发展依赖于计算机技术的进步，同时仿真技术的需求也促进了计算机技术的进步。于是，突破传统的冯·诺依曼体系结构的计算机发展起来，最早发展的是多机并行处理仿真技术。随着网络技术的发展，分布式仿真技术日益得到广泛的重视。

11.2 并行仿真

11.2.1 并行仿真的基本概念

并行仿真基于并行计算机技术。并行计算机是相对于串行计算机而言的。串行计算机就是只有单个处理器且顺序执行指令的计算机，也称为顺序计算机。顺序计算机最早从位串行操作到字并行操作，从定点运算到浮点运算逐步改进，然后逐步演变出各种并行计算机

系统。图 11.1 给出了并行计算机的发展历程。

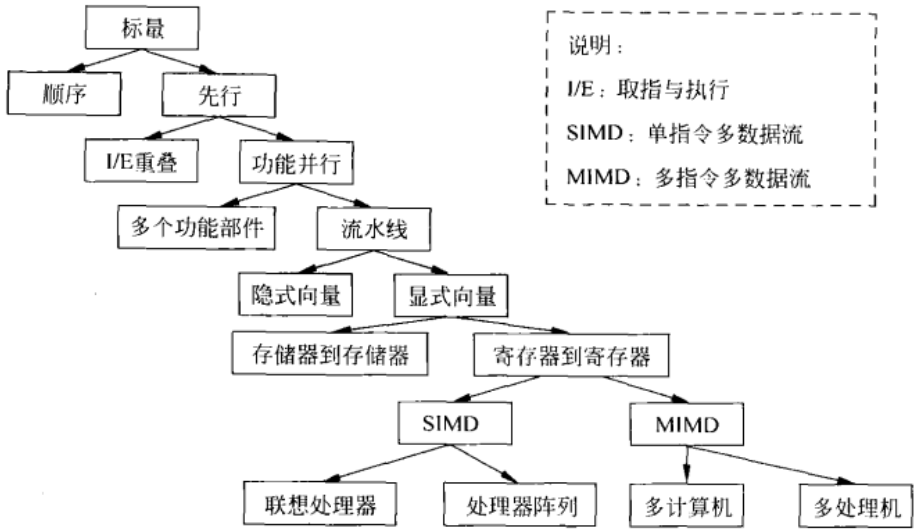


图 11.1 并行计算机的发展历程

并行计算技术首先从标量开始，从顺序标量处理(scalar processing)计算机开始，最初用先行(look-ahead)技术预取指令，达到重叠操作，进而实现功能并行。功能并行有两种方法，一种是采用多功能部件，另一种是流水线技术。后者对处理向量十分方便，从而产生了向量处理(vector processing)计算机，也称向量流水线计算机。除了时间上并行之外，进一步发展出空间上并行技术。单指令多数据流(single instruction multiply data，SIMD)系统与多指令多数据流(multiply instruction multiply data，MIMD)是典型结构。SIMD 并行机采用一个控制器同步控制所有处理器，而 MIMD 系统则采用多个控制器异步控制各处理单元。如果各处理单元通过公共存储器的共享变量实现相互通信，则称为多处理机(multiprocessors)，如果处理单元之间使用消息传递方式来实现相互通信，则称为多计算机(multi-computers)，这是当今最流行的并行计算机。

与传统的单一 CPU 系统相比，多机并行处理系统的突出优点是可极大地提高处理速度，它可以突破单机系统的速度极限(理论上，可以是无限的，实际上，由于通信和协调上的困难，仍存在新的极限)。多机并行处理系统在相当程度上突破了经典的冯·诺依曼计算机的体系，从而使得计算机不仅可以解决一般复杂性的任务，而且为解决诸如人类的思维过程等日益复杂的任务提供了可能。

并行处理技术的研究开始于 20 世纪 60 年代初期。第一台多机并行处理的计算机 ILLIAC-Ⅳ，是由美国伊利诺伊(Illinois)大学研制成功的。当时，计算机还处于晶体管年代，每一部处理机不仅体积庞大，而且造价昂贵。20 世纪 70 年代以来，大规模集成电路的迅速发展，大大加速了多机并行处理系统的研究开发工作。到 20 世纪 80 年代中期，多处理机系统技术日趋完善，表 11.1 列出了这一阶段有代表性的成果。

表 11.1 早期多机并行处理系统

型号	研制单位	公布时间	字长	存储容量	体系特征	处理速度
ILLIAC-Ⅳ	Illinois	1967 年	32/64	2KB/pe	阵列机	加法 500μs/pe 乘法 700μs/pe
PEPE	Burroughs	1972 年	32	1KB/pe	线性阵列	加法 300μs/pe 乘法 1.9μs/pe
STAR-100	CDC	1972 年	32/64	1MB (64bit)	流水线 功能并行	矢量 50～100MFlops/s 标量 5～15MFlops/s
ASC	Texas Instrument	1974 年	32	8MB (64bit)	多流水线 并行	矢量 30～100MFlops/s 标量 5～15MFlops/s
AP-120B	FPS	1975 年	32	32KB～ 1MB	流水线 功能并行	12MFlops/s
CRAY-1	Cray Research	1976 年	16/32	1MB (64bit)	专用功能 部件并行	矢量 60～100MFlops/s 标量 20～30MFlops/s
IBM 3838	IBM	1977 年	32	216KB	流水线 功能并行	30MFlops/s
MATP	Data West		32	65KB	多流水线 并行	120MFlops/s
BST	Burroughs	1978 年	48	500KB～ 8MB	阵列 流水线	矢量 50MFlops/s 标量 1.5MFlops/s
DAP	ICL	1978 年	32	2MB	阵列连接	100 MFlops/s
AD-10	ADI	1978 年	16/48	1MB	功能并行	定点 30MOP/s
VAX 8800	DEC	1983 年	32/64	2MB	功能并行	11.2MIPS
PE 3230	Perkinelmer	1984 年	32	2MB	功能并行	1.9～5MIPS
PE 3880	Perkinelmer	1985 年	32	2～16MB	功能并行	6～80MIPS

自 20 世纪 80 年代中期以来，大规模集成电路技术取得了突飞猛进的发展，计算机的内存容量、CPU 的处理速度以人们意料不到的速度在增加，而并行处理计算机也在同步地发展。

多机并行处理系统带动了并行仿真技术的发展。将多机并行处理系统用于仿真，从而形成了多机并行仿真系统，并行仿真技术由此发展起来。多机并行仿真系统系是指两个以上的独立处理机同时(simultaneous)或并发(concurrent)地执行两个以上的程序，以达到高速完成同一仿真任务的计算机系统。而采用多机并行仿真系统进行仿真的技术称为并行仿真技术。

根据多机并行仿真系统的结构、并行级别、软件体系等特征，可以将其进行如下分类。

(1) 按系统结构分类

根据多处理机构成系统时的耦合紧密程度，可以将多机并行处理系统分为：紧耦合系统、松连接系统、分布式结构系统。

紧耦合(tightly coupled)系统中所有处理机共用一个存储系统，系统内所有处理机均能平等地占用所有硬件资源，由统一的操作系统协调控制各处理机的操作同步和信息交换。这是真正典型的并行处理系统，如美国并行计算机公司生产的 PE3200MPS，SGI 公司生产的 ONYX 系统等。

松连接(loosely coupled)系统中每一个处理机均拥有自己的内存、接口、外设及操作系统,这一点与分布式结构系统相同,但机间通信采用高速总线交换信息和共享外存。各类 cluster 系统是一种典型结构。

分布式(distributed)结构系统是指构成系统的每一个处理机均拥有自己的存储器、接口、外设及操作系统,机间通信通过计算机网络或一般接口来实现。典型的是近些年来得到迅速发展的分布交互式仿真(distributed interactive simulation, DIS)系统。

(2) 按并行级别分类

根据级别,可分为 CPU 级、处理机级、计算机级三类。

CPU 级并行系统是单处理器的功能扩充,增加协调处理器、浮点处理器、函数运算器等,所以也称为多处理器系统(multi-processor system)。

处理机级并行系统最典型的是阵列处理机系统(array processor system)。各并行处理单元是包含存储器和 I/O 接口的计算器。处理器排成阵列,在同一控制器控制下执行相同的指令。它最适用于数组和矩阵的运算。

计算机级并行系统又称为多计算机系统(multi-computer system)各并行工作单元是一台包括处理器、存储器、外部设备的完整的计算机,有自己独立的操作系统,各自执行一个完整的任务,各单元可独立工作。

(3) 按指令/数据体系分类

一般的计算机执行时,指令和数据是一条一条执行的,所以也称为单指令单数据流(SISD)系统。并行仿真系统在指令执行级分为三类,即单指令多数据流(SIMD)系统、多指令单数据流(MISD)系统和多指令多数据流(MIMD)系统。

SISD 系统即各处理机同时执行同一指令,但处理结果(数据流)却采用各自的通道,前述的阵列处理机属于此种类型。

SIMD 系统的数据通过同一通道依次流过每个单元,各单元完成各自的工作,共同完成某一任务,这种系统也称为流水线处理机。

MISD 系统的各处理机一般担任一个相对完整的工作,相互之间在某些时刻有信息联系(有些还要求同步),这是功能并行处理系统。

MIMD 系统有多种结构,包括并行向量处理机(parallel vector processor, PVP),如 Cray C-90、Cray T-90、NEC SX4 以及我国的银河 1 号等;对称多处理机(symmetrical multiprocessing, SMP),如 IBM R50、SGI Power Challenge、DEC Alpha 服务器 840 和我国的曙光 1 号等;大规模并行处理机(massively parallel processing, MPP),如 Intel Paragon、IBM SP2、Intel TFLOPS 和我国的曙光-1 000 等;工作站机群(cluster of workstation, COW),如 Berkeley NOW、Alpha Farm、Digital Trucluster 等;分布式共享存储(distributed shared memory)多处理机,如 Stanford Dash、Cray T3D 和 SGI/Cray Origin 2000 等。

尽管并行处理系统有各种不同的结构,因而产生了各种不同的并行处理技术,归纳起来,下述问题是所有并行处理系统中都必须加以解决的:

① 多机并行协调控制

多机并行协调的主要特征是并行运算和资源共享。随着处理机数目的增加,对资源的竞争以及由此产生"死锁"和"排斥"问题日益严重。为解决这些问题,多机协调控制技术的有效性及完备性需要解决。目前,"时序调度法"、"优先级调度法"、"超时自限法"、"开锁关

锁法”等方法在不同类型的并行处理系统中得到应用。

② 机间同步及通信

多机并行完成同一任务，机间要频繁地交换数据，而且往往需要同步。高速地完成多处理机之间的有效通信，减少通信等待时间，确保必须的同步，一方面在硬件设计时就需要加以充分的考虑，在算法及软件实现方面也要提供有力的支持。

③ 并行算法与并行语言

为了适应多处理机并行计算的要求，必须有相应的并行算法，并要有并行语言加以实现。

④ 并行操作系统

并行操作系统除了要完成系统资源管理外，还需进行任务分解分配、多机调度、同步及通信、解决总线等资源竞争等，还能支持并行算法与并行语言，从而为用户提供实现多机并行处理的环境。

11.2.2 并行处理原理及并行仿真算法

1. 并行处理原理

显然，不同结构的并行处理系统其并行处理原理是不相同的。本节主要从指令一级介绍一些典型并行方法作为并行处理原理的简要说明。

为了便于说明，以矩阵加法为例，考虑如下矩阵加法

```
      Do   10   I = 1,N
10          C(I) = A(I) + B(I)
      Continue
```

这是一个典型的用循环语句实现矩阵加法(FORTRAN 语言)。在单指令单数据流的一般计算机中，设执行一次浮点加法需要经过“对阶”，“相加”，“规格化”，“循环”这样 $m=4$ 个时钟周期 T，N 次加法需要 $t_s=N\cdot mT$(设忽略循环计数器的操作时间)。而在单指令多数据流的阵列处理机系统中，设有 n 个处理机，控制单元对这类循环处理程序先进行判断，设 N 为 n 的整数倍，则将前 n 对 $A(I)$，$B(I)$ 同时进行相加，加法完成后同时送到 $C(I)$，然后进行下一轮。如图 11.2 所示。经过 N/n 次后完成全部加法。全部加法完成需要的时间为

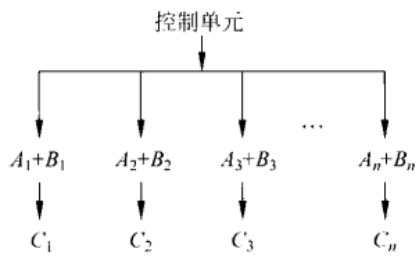


图 11.2 SIMD 并行处理原理

$$t_\Lambda=\left(\frac{N}{n}\right)mT$$

这意味着，它比单 CPU 处理机快 $n-1$ 倍。如果 N 不是 n 的整数倍，则最后一次计算时并不是所有处理机都工作，效率略有降低。

而对于 MISD 的流水线处理机来说，则将其基本操作按级进行，各处理器的级别与操作要求相对应，上一级的输出即为下一级的输入，在每个时钟周期结束时送数。仍以上述数组

加法为例，每个加法运算分为 4 个基本操作，从而分为 4 级。为便于说明，假设有 4 个处理机，分别对应一级；并设循环次数 $N=6$，则流水线处理机完成 6 次加法的步骤如图 11.3 所示。

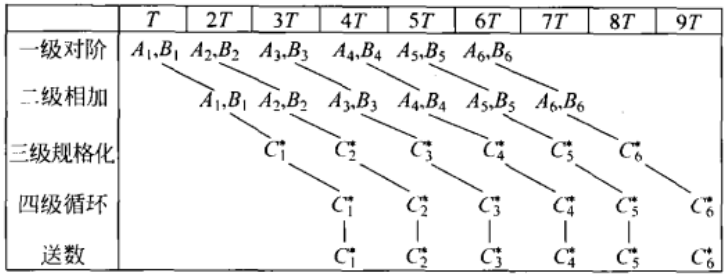


图 11.3 MISD 并行处理原理

从图 11.3 中可以看出，循环 $N=6$ 次，分为 $K=4$ 级，总的处理时间为

$$t_p = mT + (N-1)\frac{m}{K}T = 4T + (6-1) \times \frac{4}{4}T = 9T$$

当 $N \gg 1$，$t_p = \left(1+\frac{N}{K}\right)mT$。从而，与 SISD 相比，可得

$$\frac{t_s}{t_p} = \frac{NmT}{\left(1+\frac{N}{K}\right)mT} = \frac{KN}{K+N} \tag{11.1}$$

显然，当 $K \uparrow$，即级数越多，$\frac{N}{K} \downarrow$，即减少，从而 $t_s/t_p \uparrow$，即速度上升得越多。

我们还可以分析 MIMD 的执行过程，但相对来说要稍微复杂一些，读者从上面的讨论中不难看出其效率的提高更为显著。

2. 并行仿真算法

并行仿真算法将在任务一级进行讨论。其基本问题主要表现在两方面：一是如何进行任务分配，即根据任务的特点、计算量和仿真的要求，考虑并行处理机的数量、结构，将一个任务分解成若干子任务，每个子任务交由相应的处理机进行处理；二是如何保证仿真过程中数据的同步并尽可能减少通信量。显然，对应于不同结构的并行处理机系统，并行算法也就很不相同，同时，被仿真的对象的特点也极大地影响着仿真算法的实现。

本节我们将讨论两类典型对象的并行仿真算法：多项式计算、矢量乘法。下一节将重点讨论微分方程并行求解。这些是并行仿真中最常遇见的。

在讨论这些算法时，并行处理机的结构考虑为：m 台独立处理机，它们之间可以相互通信，通信方式可以是单总线系统(图 11.4)，也可以是多总线系统(图 11.5)。

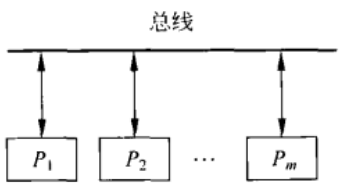


图 11.4 单总线系统

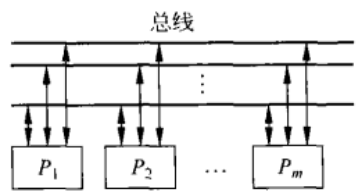


图 11.5 多总线系统

(1) 多项式计算

多项式一般可用如下表达式描述

$$y=\sum_{i=0}^{n}C_ix^i=C_0+\sum_{i=1}^{n}C_ix^i=C_0+\sum_{i=1}^{n}Z_i \tag{11.2}$$

其中，$Z_i=C_ix^i(i=1,2,\cdots,n)$。设 $n=8,m=8$，则并行算法可描述如下(参见图 11.6)。

处理器编号		0	1	2	3	4	5	6	7
①		x	x	x	x	x	x	x	x
②	第 1 步	x	x^2	x^2	x^2	x^2	x^2	x^2	x^2
	第 2 步	x	x^2	x^3	x^4	x^4	x^4	x^4	x^4
	第 3 步	x	x^2	x^3	x^4	x^5	x^6	x^7	x^8
③		Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8
④	第 1 步	$\sum_{i=0}^{1}Z_i$	$\sum_{i=1}^{2}Z_i$	$\sum_{i=2}^{3}Z_i$	$\sum_{i=3}^{4}Z_i$	$\sum_{i=4}^{5}Z_i$	$\sum_{i=5}^{6}Z_i$	$\sum_{i=6}^{7}Z_i$	$\sum_{i=7}^{8}Z_i$
	第 2 步	$\sum_{i=0}^{1}Z_i$	$\sum_{i=1}^{2}Z_i$	$\sum_{i=1}^{3}Z_i$	$\sum_{i=1}^{4}Z_i$	$\sum_{i=2}^{5}Z_i$	$\sum_{i=3}^{6}Z_i$	$\sum_{i=4}^{7}Z_i$	$\sum_{i=5}^{8}Z_i$
	第 3 步	$\sum_{i=0}^{1}Z_i$	$\sum_{i=1}^{2}Z_i$	$\sum_{i=1}^{3}Z_i$	$\sum_{i=1}^{4}Z_i$	$\sum_{i=1}^{5}Z_i$	$\sum_{i=1}^{6}Z_i$	$\sum_{i=1}^{7}Z_i$	$\sum_{i=1}^{8}Z_i$
⑤									y

图 11.6 多项式并行计算

① 将 x 送入寄存器，同时输入处理器(编号从 0 到 7)。

② 同时按以下规则执行乘法求 $x^1,x^2,\cdots,x^8$：在第 j 步，对编号为 2^{j-1} 的处理器与其左边的第 2^{j-1} 个处理器中的数相乘，从编号为 2^{j-1} 的处理器到编号为 $m-1$ 的处理器同时进行，即：

第 1 步 $2^{j-1}=2^0=1$，第 1 号到第 7 号处理器同时与其左边第 1 个处理器的数相乘：$x\times x=x^2$；

第 2 步 $2^{j-1}=2^1=2$，第 2 号到第 7 号处理器同时与其左边第 2 个处理器的数相乘：$x^2\times x=x^3$(第 2 号)，$x^2\times x^2=x^4$(第 3 号到第 7 号)；

第 3 步 $2^{j-1}=2^2=4$，第 4 号到第 7 号处理器同时与其左边第 4 个处理器的数相乘：$x^4\times x=x^5$(第 4 号)，$x^4\times x^2=x^6$(第 5 号)，$x^4\times x^3=x^7$(第 6 号)，$x^4\times x^4=x^8$(第 7 号)。

③ 并行乘法：$Z_i=C_ix^i(i=1,2,\cdots,n)$分别同时在第 0 号到第 7 号处理器中进行。

④ 并行加法：编号为 2^{j-1} 的处理器与其左边的第 2^{j-1} 个处理器中的数相加得到 $\sum_{i=1}^{n}Z_i(i=1,2,\cdots,n)$，用三步可以完成。

⑤ 计算 $y=C_0+\sum_{i=1}^{n}Z_i$。

当 $n>m$ 时，上述算法需要适当修改。

(2) 矢量乘法

上一节我们介绍了矢量加法，本节讨论矢量乘法。考虑如下表达式

$$\boldsymbol{y}=\boldsymbol{A}\boldsymbol{x},\quad \boldsymbol{y}=(y_1,y_2,\cdots,y_n)^{\mathrm{T}},\quad \boldsymbol{x}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$$

其中，$\boldsymbol{A}$ 为 $n\times n$ 矩阵

$$\boldsymbol{A}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{mn} \end{bmatrix}$$

当处理机的个数 m 等于矢量的维数 n 时，算法十分简单，即每个处理机负责一个矢量的计算，而每个处理机内部仍按串行计算。如果 m 大于 n，有两种策略，称为垂直策略和横向策略。

① 垂直策略

垂直策略的基本思想是：将 n 个矢量分配给每个处理机计算，剩余 $m-n$ 个处理机空闲。此时的利用率为 n/m。例如，如果 $n=5$，$m=8$，利用率为 $5/8=0.625$。

② 横向策略

垂直策略算法简单，从而易于实现，但处理机的利用率较低。为提高处理机的利用率，人们提出了横向策略，下面以 $m=8$，$n=5$ 来加以说明。对每个矢量，其计算公式为

$$y_i = a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{i5}x_5 \quad (i = 1,2,\cdots,5)$$

图 11.7 给出了并行算法的实现过程。其中第 0 号处理机主要用于 y_1 的计算，第 1 号处理机用于 y_2 的计算，依次类推。第 5，6，7 号处理机承担部分功能。

由图中不难计算出处理机的利用率。图中带“－”的块表示处理机为闲（共 19 块），总块数表示所有处理机可用时间（共 64 块），则利用率为 $(64-19)/64=0.703$，较垂直算法提高了 8%左右。显然，横向算法的控制要复杂得多，而且还需要交换数据。因此，到底采用哪一种方法为宜，要根据 m 与 n 的大小具体分析。

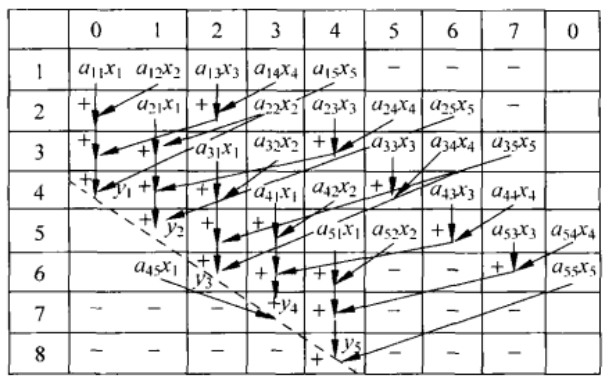


图 11.7 矢量乘法横向算法原理图

如果 $n>m$，可先采用垂直策略，计算出 m 个分量，计算若干组（如果有的话）后，剩余的分量再根据情况决定到底是采用垂直算法还是横向算法。

11.2.3 面向微分方程的并行仿真

本节我们讨论微分方程的并行计算问题。算法的基础仍然是单处理机采用的各种方

法。读者从前面的各章讨论中对这些方法已有清楚的了解，本节以四阶 Adams 方法为例说明当采用这些方法用于并行计算时需要重点解决的问题。设方程为$\dot{Y}=F(Y,t)$，$Y(0)=Y(t_0)$。若先采用四阶 Adams 显式进行预报，然后用四阶 Adams 隐式加以校正，则有

$$\begin{cases} Y_{i+1}^{(0)} = Y_i + \dfrac{h}{24}(55F_i - 59F_{i-1} + 37F_{i-2} - 9F_{i-3}) \\ Y_{i+1} = Y_i + \dfrac{h}{24}(9F_{i+1}^{(0)} + 19F_i - 5F_{i-1} + F_{i-2}) \end{cases} \tag{11.3}$$

在串行情况下，若方程为 N 阶，设计算 $Y_{i+1}^{(0)}$ 的时间为 Nt_{lc}，计算 $F_{i+1}^{(0)}$ 的时间为 t_{f}，计算 Y_{i+1} 的时间也为 Nt_{lc}（因为其计算表达式基本相同），而为下一步准备需计算 F_{i+1}，计算时间与 $F_{i+1}^{(0)}$ 相等为 t_{f}，总时间为 $t_{\mathrm{s}}=2(Nt_{\mathrm{lc}}+t_{\mathrm{f}})$。

现采用 M 个处理机并行计算，设它们相互均有通信能力，下面介绍两种基本算法。

1. 方程分段法

方程分段法的基本思想是：将 N 个微分方程分成 M 组子方程组，每个子方程组由一个处理机负责处理，每步要交换数据。设 $M=2$，原方程组分为两个子集，包括 N_1，N_2 个方程（$N_1+N_2=N$），并记 $N_{\max}=\max(N_1,N_2)$。又设两子集之间的耦合系数为 $k(0\leqslant k\leqslant 1)$，它直接决定了子集之间的通信与交换数据所需要的时间。

一般说来，即使采用同一方法计算，各子集在步内的计算时间也是不可能完全相等的。设处理机 P_1 负责 N_1 个方程子集的计算，处理机 P_2 负责 N_2 个方程子集的计算，还假设前者每步计算所需时间小于后者所需时间。图 11.8 表示某一步两个处理机计算过程及两者之间的时序关系。

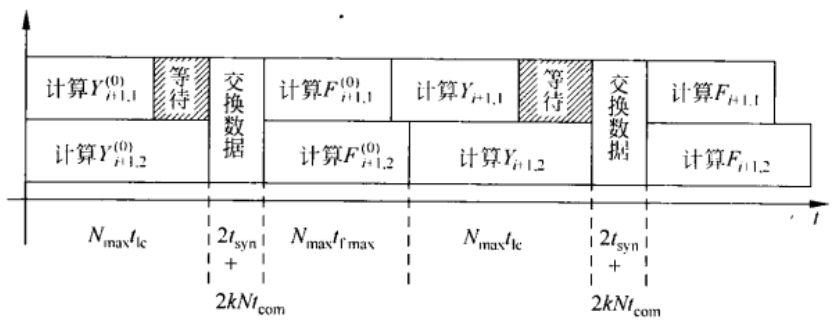


图 11.8 方程分段法并行仿真

由于两个子集之间有耦合，所以要求保持同步，以便交换信息。例如，当处理机 P_1 完成 $Y_{i+1,1}^{(0)}$ 的计算时，由于其速度快，处理机 P_2 尚未完成 $Y_{i+1,2}^{(0)}$，所以它必须等待。当处理机 P_2 完成 $Y_{i+1,2}^{(0)}$ 时，两者交换数据，以便分别计算 $F_{i+1,1}^{(0)}$，$F_{i+1,2}^{(0)}$，继而分别计算 $Y_{i+1,1}$，$Y_{i+1,2}$。同样，处理机 P_1 完成其计算后需要等待，直到 P_2 完成，然后交换数据，以便各自计算 $F_{i+1,1}$，$F_{i+1,2}$，然后进入下一步。这种在一步计算中，分成若干段同步，交换信息，因而称为方程分段法。

若将 P_2 完成 $Y_{i+1,2}^{(0)}$ 和 $Y_{i+1,2}$ 的时间近似相等，记为 $N_{\max}t_{\mathrm{lc}}$，P_1 与 P_2 的同步时间记为 $2t_{\mathrm{syn}}$，P_2 计算 $F_{i+1,2}^{(0)}$ 与 $F_{i+1,2}$ 的时间也近似相等，记为 t_{fmax}，则可计算出双处理机采用方程分段法并行计算时每步所需时间为

$$t_p = 2(N_{max}t_{lc} + t_{fmax}) + 4t_{syn} + 2kNt_{com} \tag{11.4}$$

其中最后一项为因耦合而需交换数据的时间，它不但与方程的阶数有关，更与耦合系数有关。下面，我们进一步比较 t_p 与串行计算时间 t_s。显然，t_p 与任务分配的均衡度有关，均衡度包括方程分配均衡度 D_N 和算法安排的均衡度 D_f。它们由下式定义

$$D_N = \frac{2N_{max} - N}{N}, \quad D_f = \frac{2t_{fmax} - t_f}{t_f} \tag{11.5}$$

当 $D_N=0, D_f=0$ 时

$$t_p = \frac{1}{2}t_s + 4t_{syn} + 2kNt_{com} \tag{11.6}$$

进一步，若两子集独立，则 $k=0$，从而 $2kNt_{com}=0, t_{syn}=0$，此时 $t_p=\frac{1}{2}t_s$，这是显而易见的。

方程分段法原理简单，易于实现，其效率取决于任务分配的均衡度，一方面要降低子集之间的耦合度，另一方面要考虑各子集计算任务量的均衡度。

2. 并行预测校正法

方程分段法算法虽然简单，但子集任务量的均衡难以掌握，特别是难以兼顾耦合度和均衡度两方面的要求。并行预测校正法基于预测与校正的计算量基本相近，采用一组处理机专用于预测，另一组处理机专用于校正，任务均衡度易于保证，这样，处理机空等的时间可大大减少，而且只需要在每步结束时才交换数据，减少了同步及交换数据的次数，这样，并行处理的效率就可得以提高。下面仍以双机为例，说明这种算法的执行过程(参见图 11.9)。

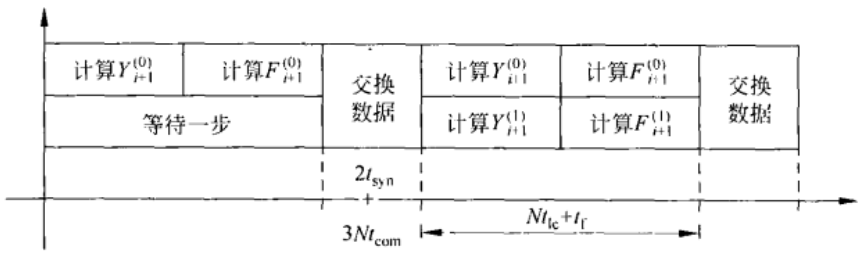


图 11.9 并行预测校正法并行仿真

从图 11.9 中可以看出，假设由处理机 P_1 负责预测，它首先计算 $Y_{i+1}^{(0)}$，然后根据 $Y_{i+1}^{(0)}$ 计算出 $F_{i+1}^{(0)}$，这些计算均不需由 P_2 读入数据。处理机 P_2 负责校正，同样，它先计算 $Y_{i+1}^{(1)}$，然后计算出 $F_{i+1}^{(1)}$。这里首先遇到处理机 P_2 的启动问题，因为在第一步，处理机 P_1 刚进行预报，其 $Y_{i+1}^{(0)}$、$F_{i+1}^{(0)}$ 尚未计算出来，P_2 无法同时启动校正。为此，P_2 需滞后一步，P_1 第一步预报结束后，交换数据，从第二步起 P_2 进行校正；而此时 P_1 只能用本身预报值作为校正值再进行预报。这意味着，整个仿真结果滞后一步。

下面进一步分析并行预测校正法的执行时间。显然，无论是 P_1 预测还是 P_2 校正，其执行时间可按 Nt_{lc} 考虑，计算结束时两机同步时间为 $2t_{syn}$，交换数据需要的时间为 $3Nt_{com}$，用于交换 $F_{i+1}^{(0)}$(由 P_1 送至 P_2)、Y_{i+1}、F_{i+1}(由 P_2 送至 P_1 以取代 Y_i, F_i)，因此，可以得到

$$t_p = Nt_{lc} + t_f + 3Nt_{com} + 2t_{syn} \tag{11.7}$$

综上所述，并行预测校正法机间同步易于实现，无用开销时间少，通信量与系统内部的

耦合程度无关；其缺点是，仿真结果滞后一步，而且当处理机的数目是奇数时，至少有一台处理机要安排作为他用。当用两台处理机并行计算时，该方法的极限速度是其计算时间不超过单机计算时间的二分之一。

11.2.4 并行仿真任务划分

以往的仿真方法一般是针对串行计算机设计的，在并行计算机上难以直接应用，因此必须设计新的并行算法。并行算法的基本思想是将求解的整个计算分割成若干个可以解耦的独立求解过程或向量运算块。通常有三种基本构造途径：

(1) 系统(或问题)分割并行化方法

系统分割是将一个大系统分割成若干个子系统，进行解耦处理，并把这些子系统分配给不同的处理机进行并行计算。这种分割又称为问题分割或空间分割。它适用于大系统数值仿真，而且使用显式公式比较方便、有效。主要问题是各处理机的负载分配平衡和通信问题。

(2) 方法分割的并行化方法

方法分割是利用方法内在的并行性，实质上是在不同的处理机上同时进行方法不同部分的值的计算，特别是在不同的处理机上同时执行方法不同时间点上的值的计算。它适合于具有较少处理机的并行计算机使用，因而可独立的任务个数有限。

(3) 时间分割的并行化方法

时间分割与逐步积分的概念恰好相反，它可同时完成许多步的积分，即在时间轴上可同时计算许多点解的近似值，但同步、通信的开销会比较大。

上述方法归根结底是任务划分问题。任务划分是指把一个计算问题分解为多个可并行执行的任务。为了获得最大并行度，希望划分后任务粒度(即每个任务的计算量大小)很细，能以并行执行的任务很多，因而可采用更多的处理机。在单处理机系统中，任务粒度最粗，即没有任何并行性可言。在数据流计算机中，任务粒度最细，并行性最大。但是，太细的粒度会造成通信、同步控制开销大。

本节以 MIMD 计算机为例，讨论并行仿真的任务划分问题。对于同构型 MIMD 机器来说，若用串行积分方法求解状态方程，则任务划分方法可分为三类：算术运算级、程序方程级和状态变量级。

在算术运算级的划分中，把仿真计算模型的每个四则运算，三角函数运算或积分运算作为一个任务，用数据流的分析方法构造编程任务图。对于用面向方程的仿真语言或 FORTRAN 等高级语言编写的程序，由用户或系统软件逐个分析每一个程序方程(赋值语句)，按照方程的算术运算和数据依赖关系来生成任务图。对于用方框图描述的仿真计算模型，则直接把每一个方框算子作为一个任务。把方框的连线作为任务图的有向边。以著名的范德波尔(VandePol)方程为例

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} - u\frac{\mathrm{d}x}{\mathrm{d}t}(x^2 - 1) + x = 0$$

化为状态方程，有

$$\begin{cases}\dot{x}_1 = x_2 \\ \dot{x}_2 = u(x_1^2 - 1)x_2 - x_1\end{cases}$$

其方框图如图 11.10 所示。根据此方框图可直接得到如图 11.11 所示的任务图。

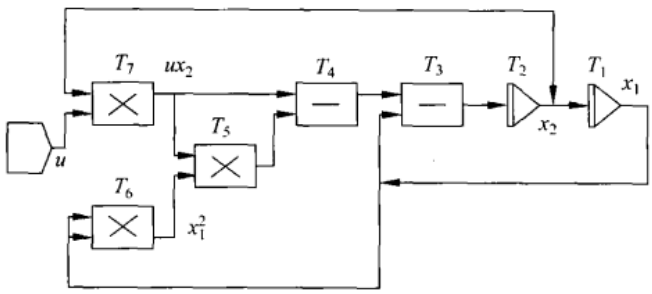


图 11.10 范德波尔方程方框图

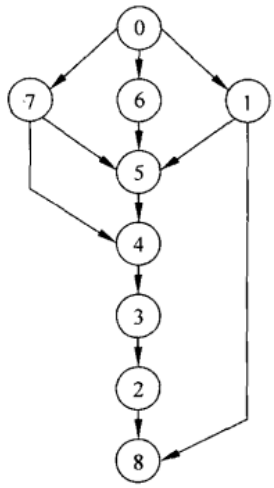


图 11.11 范德波尔方程任务图

具体做法是：去掉方框图中积分算子的输出端连线，然后为所有没有前趋的任务节点构造一个公共虚拟前趋，为所有没有后继的任务节点构造一个公共虚拟后继。由于忽略不计通信开销，故每轮迭代结束后的状态变量交换未在任务图中反映。运算级的划分，任务粒度较细，当状态方程较少时，仍然容易实现处理机的负载平衡。但同时，任务间通信较多，故适合于紧耦合的多处理机系统。

程序方程级的划分方法把仿真程序中的每一个程序方程作为一个任务，根据方程变量的依赖关系来构造编程任务图。仍以范德波尔方程为例，若采用四阶龙格-库塔法，则对应的程序如下，其任务图如图 11.12 所示。

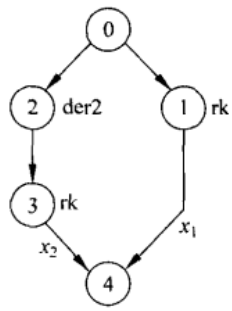


图 11.12 程序方程级的范德波尔任务图

```
While time<maxtime do
begin
    for 1 :=1 to 4 do
        begin
            der1 :=x2
            der2 :=u*(1-x1*x1)x2-x1
            x1 :=rk(der1,i,1)
            x2 :=rk(der2,i,2)
        end
    time :=time+h
end;
```

虽然方程级划分的任务粒度比运算级的较粗，但任务间仍有编程约束，任务间通信仍然较多，故这种划分方法也常用于多处理机系统。

在状态变量级的划分中，每个状态方程被作为一个任务。在一轮迭代中任务间没有数

据依赖关系，每一轮迭代结束后各任务交换一次状态变量。这种方法的任务粒度比前两种方法的都粗，任务间无编程约束，且通信较少，故不仅适用于多处理机系统，还适用于多计算机系统。另外，这种方法的任务划分，用户手工实现起来非常容易。变量级划分方法的缺点是，处理机负载可能会很不平衡。例如，每一个处理机分配到一个状态方程，若其中某个处理机中右端函数的计算量很大，则该处理机就将成为系统的瓶颈。

11.2.5 并行仿真的实现

1. 并行计算操作系统

操作系统是控制和管理计算机系统的硬件和软件资源，合理地组织计算机工作流程以及方便用户的程序集合。对于计算机而言，操作系统是至关重要的。在并行仿真系统中，事件具有并发性，因此在系统运行过程中，就会出现多个程序向同一资源提出申请，或是某一资源长时间空闲等情况，这些都是系统所不愿意遇到的，然而恰恰又是经常出现的情况。并行的操作系统就是为了在多个处理器和其他资源之间进行仿真任务的协调，保证系统可以正常有序地进行并行处理。并行仿真要求系统中所有的事件必须按照它们的因果关系顺序执行，因果关系主要是由每个事件的时间戳决定。现在已经出现了可以完全支持并行仿真的操作系统如 time warp operating system 和 mimdix system。此类操作系统对仿真编程语言没有特别要求，仿真过程中的策略和仿真调度都由操作系统完成，但是，如果采用此类操作系统，在仿真建模和仿真运行过程中对于一些影响仿真性能的策略和参数无法进行调整和修正，因此有时候仿真效果不是很理想。

2. 并行仿真同步策略

并行仿真系统必须根据系统内事件发生的先后顺序执行事件，所以只有确认各逻辑过程 LP(logical process)中不会出现比当前事件时间戳小的事件后才能安全地执行当前事件，这样的要求被称为因果关系约束，它是有效执行并行仿真所必须满足的必要条件。

由于大多数的并行仿真模型是异步的，具有不规则、随机的事件发生特点或数据依赖关系，因果关系约束的满足是通过同步执行并行仿真模型中的事件实现的，这就是并行仿真的同步策略。目前，同步策略主要有四种，分别是保守策略、乐观策略、混合策略和自适应策略。

1) 保守策略

保守策略严格禁止发生因果关系错误，保证事件按时间发生的先后顺序在并行系统中执行。Chandy-Misra 在 1979 年提出的保守策略是最典型的一种，该策略限定，各个 LP 之间传递消息的拓扑结构在运行模型前已经确定，而且消息按时间戳顺序到达 LP 的接收通道。这种限定保证了每个 LP 最近收到的消息时间戳是后续消息的下限。LP 所接收到的消息按 FIFO 顺序进行处理，每个 LP 都有一个局部虚拟时间(local virtual time，LVT)，LVT 等于 LP 刚刚处理或正处理的消息时间戳。仿真过程重复选择 LVT 最小的 LP，如果有任务需要处理，则执行；如果没有任务，寻找下一个需要处理的 LP，并将此 LVT 推进。保守策略保证了每个过程按照时间戳非递减顺序执行事件。

虽然这种策略能够避免因果关系错误，但是却容易发生死锁现象。解决死锁现象的方法有两种，死锁避免法和死锁恢复法。其中采用零消息(null message)避免死锁是较常用的一种。零消息本身只携带一个时间戳，对仿真系统的活动没有影响，只是表示该 LP 不会发出时间戳小于零消息时间戳的消息。

现在对于零消息的发送有三种计算方法：

(1) 普通零消息法：该算法由三部分组成，①每个 LP 计算其最小输入时间(earliest input time，EIT)，也就是它所有输入源 LP 的最小输出时间(earliest output time，EOT)的最小值；②处理时间戳小于 EIT 的所有消息；③如果 LP 在收到一个零消息或处理一个消息后，导致其 EOT 发生变化，则向所有输出源发送带有更改后的 EOT 的零消息。零消息可以单独发送，也可以捆绑在一个常规消息上一起发送。

(2) 条件事件法：该算法与普通零消息法大致相同，只是在计算 EIT 的时候有区别。条件事件法定义了一个条件输出时间(conditional output time，COT)，它等于 LP 中未处理的最小时间戳事件的时间与超前量之和。每个 LP 都计算自身的 COT，仿真系统将它们集合起来，得出最小的条件输出时间(earliest conditional output time，ECOT)，ECOT 就是系统中所有 LP 的 EIT。

(3) 加速零消息法：该算法的含义是在零消息法中加入了 ECOT 的计算，也就是前两种方法的结合。ECOT 的初始计算由一个空闲的 LP 完成，在仿真运行过程中，LP 在推进 EIT 的同时计算 ECOT，那么 LP 推进的 EIT 则是两种计算结果的最大值。

2) 乐观策略

乐观策略允许 LP 可以按照任何顺序执行事件，如果发生因果关系错误，则通过回退方法重新执行仿真系统，以此来纠正违反因果关系错误所产生的后果。在这种策略中，每个 LP 有一个 LVT，用来标记该 LP 对消息的执行情况，LP 之间通过带有时间戳的消息进行通信。如果 LP 收到滞后的消息，则表明发生了因果关系错误，仿真系统需要回退到错误发生以前的位置，重新执行系统。因此采用乐观策略的仿真系统必须具备两个特殊功能：(1)删除错误的消息及其执行结果；(2)系统中保存了以前执行事件的所有信息和系统的状态变化。这就涉及一个问题，发生因果关系错误时系统需要回退到什么位置。如果回退的内容很多，则需要大量的空间保留系统信息，这样就会影响仿真系统的性能。因此在乐观策略中应用了全局虚拟时间(global virtual time，GVT)的概念。当仿真系统发生因果关系错误需要回退时，系统首先根据部分已经执行的事件和未执行的事件时钟计算其中的最小值作为 GVT。一旦 GVT 计算出来，仿真系统只需要回退到 GVT 时刻即可。

虽然乐观策略有效地利用了系统的并行性，但是它需要保存系统状态信息，占用了大量的存储空间，因此提高乐观策略的性能主要是如何尽可能少地保留信息而不影响系统运行。

一般有两种方法实现这一点：(1)及时计算并行仿真系统中的 GVT，对所有时间戳小于 GVT 的系统状态不必保留；(2)合并系统状态和需要交互的消息，以此来提高仿真系统的性能。

3) 混合策略

通过研究比较，人们发现保守策略和乐观策略的优缺点可以互补。保守策略对仿真并行性利用不高，运行效率低，但不会发生因果关系错误；乐观策略容易发生因果关系错误，而且增加仿真运行的复杂性，但能有效地利用仿真系统的并行性。将两者结合起来，利用各

自的优点，则能更好地提高并行仿真系统的性能，这就是混合策略。

混合策略分为两类：一类是在保守策略中加入乐观策略，该策略以保守策略为基础，每个 LP 按照循环周期执行事件，在每个循环周期里要尽可能多地执行事件。每个 LP 在执行过程中都有可能产生事件，所有 LP 所产生的事件的最小时间戳作为乐观策略的执行上限，凡是超过了该上限的事件都要退回来。另一类是在乐观策略中加入保守策略，该策略是在一定的时间范围内控制乐观策略的推进，规定了乐观策略可以自由执行的上下限，防止执行时间过长所导致的过多的系统状态存储。

4）自适应策略

自适应策略实质上也是一种混合策略，但它可以随着状态的变化，动态地修改其执行方式，从而改变系统的乐观性。在仿真过程中，可以根据需要调整乐观策略自由执行的上下限，使自适应策略贴近任何一种策略。

3. 并行仿真的任务实现

为了将一个实际系统映射到仿真计算机，首先对实际系统的功能进行层次化和细致化的处理，即将整个系统的功能进行分割，使系统成为由众多相对独立的细小功能组成的一个集合；然后将每一个功能在仿真系统中映射为一个逻辑模块(LP)，仿真系统则是由以 LP 为基本模型构件所组成的集合；最后则是选择合适的仿真语言对设计好的仿真系统进行编程实现以及结果分析。

1）实际系统的层次化分解

复杂大系统的含义有两点：功能上的复杂性和结构上的复杂性，为了对此类系统进行仿真，首先对其进行层次化分解是非常必要的。不同的复杂系统有不同的分解方法，对于功能复杂的系统而言，如冷连轧动态生产过程，在分解的时候首先将其分为几个功能相对独立的子系统，规划出系统之间的关系和交互数据，然后再对每个子系统进行分解，形成功能单一的小模块。对结构上复杂的系统，如(very large scale integration，VLSI)系统，人们对该系统的仿真已经有很多的研究，对此系统采用的仿真分解方式是先将电路结构抽象成一个个层次，然后再根据各自的特点进行分解。

2）逻辑模块

逻辑模块(LP)是并行仿真系统中的基本构件，一个 LP 对应着系统中的一个实体。它由以下几个部分构成：

(1) 功能部分，每个 LP 都是一个功能模块，在并行仿真系统中完成一定的功能。

(2) 局部虚拟时间(LVT)，在仿真系统运行过程中，除了仿真系统的全局时间外，每个 LP 本身都有一个 LVT 来推进其运行，LP 可以通过消息传递了解另一个 LP 的 LVT 和事件处理情况。

(3) 将来事件表(future event list，FEL)，LP 将收到的消息进行处理后形成本地事件，如果事件的时间戳比当前的 LVT 大，则将其放入 FEL 中等候处理。

(4) 消息，它是 LP 与其他 LP 进行联系的工具，消息在 LP 之间相互传递。收到消息的 LP 必须先将其处理，然后才能形成本地所需的事件。

每个 LP 都具有输入通道和输出通道，需要输出或从外部 LP 输入的消息经过这些通道进入 LP。通道在对消息进行处理时遵循一定的规则，如先到先服务规则(first come first

served,FCFS)或先入先出规则(first in first out,FIFO)。

LP 在处理消息的时候还有几个重要的概念:

① 最早输出时间(EOT):对于一个 LP 而言,EOT 是指将来它发送消息的最早时间。

② 最早输入时间(EIT):一个 LP 的 EIT 是指它将来从其他 LP 收到消息的最早时间,因此,LP 可以处理任何一个时间戳小于 EIT 的事件而不会发生因果关系错误。

③ 超前量:它是指 LP 还要经过多长时间才发送下一条消息。

④ 消息的来源集合与去向集合(source and destination):消息的来源集合是指所有向该 LP 发送消息的 LP 的集合,消息的去向集合是指所有收到该 LP 消息的 LP 的集合。

3) 仿真编程

无论是串行仿真还是并行仿真,在建立数学模型之后,设计者必须为其选择一个合适的编程语言。此时既可以选择现成的仿真语言如 Simula,CSIM 等,也可以选择由一般语言编写的仿真库,当然也可以选择一种通用语言如 C 和 FORTRAN 等。采用仿真语言进行仿真编程的优点在于,其本身提供了仿真系统的运行方式,如时间驱动方式或事件驱动方式,而且随机数的产生以及仿真结果、报表统计等功能都变得非常容易,但是设计者必须学习一门新的语言,这需要时间和精力。如果采用通用语言进行编程,一般设计者无须学习,而且这种语言的适应范围很广,但缺点是需要对仿真系统和仿真过程进行深入的理解,因此增加了编程者的难度。

在并行仿真中,由于逻辑过程比串行仿真复杂得多,所以尽量考虑使用现有的并行工具,以减轻编程者的负担。现在常用的并行仿真语言主要有三类,一是在串行仿真语言中加入并行处理;二是开发并行库和 API 接口;三是在并行语言中加入仿真功能。仿真系统的并行实现由两点体现出来,数据共享和消息传递。消息传递方法由于其适应性强而成为大多数系统采用的方法,也有的系统采用数据共享的方式实现了并行仿真。

11.3 分布交互式仿真

分布交互式仿真(distributed interactive simulation,DIS)是 20 世纪 80 年代初产生的,它是指在高速计算机网络支持下,将分布在不同地点的仿真系统集成起来,通过仿真实体间的实时数据交换,在时空一致的人机交互仿真环境中并行进行仿真试验的一种先进仿真技术。

11.3.1 分布交互式仿真的发展

尽管在 20 世纪 80 年代初就出现了多台计算机联网仿真,并作为一种松连接的并行仿真技术加以使用,但是,随着计算机网络技术的迅速发展,网络传输速度大大提高,而多处理机系统无论其硬件技术还是其软件技术(特别是并行操作系统和并行语言)的发展却步履缓慢,加上仿真技术所面对的问题规模日益庞大,种类也日趋增多,于是人们重新着力于分布式仿真技术的研究。

分布交互式仿真起始于美国国防部高级研究计划署(DARPA)和美国陆军制定的

SIMNET 研究计划。25 年来，分布交互式仿真技术的发展可分为 4 个阶段：SIMNET 研制和使用阶段、DIS 研制和使用阶段、ALSP 阶段和高层体系结构(HLA)阶段。

1. SIMNET

20 世纪 80 年代初开始实施的 SIMNET 计划的目标是将已经开发或计划开发的分散在各地的多个仿真系统在网络环境下集成起来，以实现更为复杂的仿真任务。该计划的背景是第一次海湾战争，模拟未来的战场，演示验证实时联网的人在回路中的作战能力。

该系统以德国和欧洲中部的一个 50km×70km 的环境为作战战场，数据库内有详细的数字化地图，作战武器包括能独立仿真的 M1 主战坦克、装甲车、雷达车、A-10 对地攻击机等，通过长距离的声音和电子消息传输进行通信，并能模拟采用苏军战术的计算机生成兵力(computer generated force，CGF)。在 20 世纪 80 年代末，SIMNET 已经发展成为具有约 260 个地面车辆仿真器和飞机飞行仿真器、通信网络、指挥所和数据处理设备等互联的网络，分布在美国和德国的 11 个城市。

SIMNET 为了实现这些异构的仿真器在网络环境下的集成，采用了对象/事件的系统结构、仿真节点具有自治性、DR(dead reckoning)算法等技术成为分布交互式仿真的技术基础。

2. DIS

SIMNET 计划的成功实施，促进了仿真技术网络化的发展。1989 年 3 月，由美国 Central Florida 大学的仿真与训练研究所(institute of simulation & training，IST)主办召开了第一届 DIS 研讨会，并成立了工作组，此后每年召开两次研讨会，从而有力地推动了 DIS 的研究，加速了 DIS 的发展。1992 年在第 6 届研讨会上，美国陆军仿真训练装备司令部(simulation，training and instrumentation command，STRICOM)提出了 DIS 的结构，并着手制定 DIS 协议。

在 DIS 的发展过程中，分布交互式仿真从仅支持基于同构网络发展成为对异构网络的支持，从概念性研究发展到联合作战的人员训练、武器研制、战术训练和空中交通管制等军事与非军事领域的仿真，逐步走向实用，并产生了 DIS 2.0 标准。

DIS 2.0 标准较好地解决了“人在回路”的实时仿真，支持网络环境下广播方式交换数据(best-effort broadcast)，通过 PDU(protocol data units)实现实体状态信息的交互，包括实体的位置、姿态、速度、加速度、角速度等。为了提高通信的有效性，继续采用 SIMNET 中使用的 DR 机制。

然而，随着 DIS 需求的发展，DIS 2.0 越来越不适应要求，主要表现在以下几个方面：

(1) 应用范围：DIS 2.0 虽然具有一定的 CGF(计算机生成兵力)能力，但未能解决 CGF 的聚合与解聚问题。为此，构造仿真技术快速发展起来，聚合级仿真协议(aggregate level simulation protocol，ALSP)与 DIS 并行，但其体系结构与 DIS 完全不同，而两者的融合似乎难以实现。

(2) 交互与通信：DIS 2.0 仅支持无连接的广播方式，这种广播方式虽然能较好地满足“人在回路”的实时仿真的需要，然而不能保证无误地发送与接收，特别是以广播方式在网络上发送数据和交互信息，不管这些信息是否为其他仿真节点所需要，一方面造成网络通信拥

挤、堵塞，从而影响实时性，而且接收方必须将所有数据接收，然后进行过滤。随着系统中实体数目增加，节点间的信息交互呈指数增长，极大地制约了系统规模。广播方式不能支持由事件驱动和时间驱动的仿真，因而也难以在其他类型的仿真中得到推广。

(3) 协议的扩展：随着 DIS 应用范围不断拓展，新的仿真对象及交互作用类型相继出现，这就要求 DIS 的标准不断扩展，然而 DIS 2.0 的扩展能力有限，包括数据描述方式的多样性、数据传输的多样性等。

DIS 2.0 的上述不足，促进了新一代体系结构的产生，这就是高层体系结构(high level architecture，HLA)。关于 HLA，我们将在下一节介绍。

3. ALSP

聚合级仿真协议(ALSP)是与 DIS 2.0 并行产生的，其背景是军事仿真的需要，DARPA 委托 Mitre 公司对实验进行分析研究。Mitre 公司对照 SIMNET 对实验进行了技术分析，提出了聚合级仿真协议(ALSP)。

ALSP 的目标是使现有的多个聚合级作战仿真应用可以通过局域网或广域网相互交互。在概念上，每个仿真控制其自身对象，并且共享有关其自身与其他仿真的信息。ALSP 吸取了 SIMNET 技术中的一些原则，如没有中心节点、地理分布、自主属性和基于消息的协议等，同时发展了系列聚合级仿真所需的技术：时间管理、数据管理和体系结构等。ALSP 与 DIS 2.0 的 CGF 相比，其优点在于聚合级实体对网络带宽的要求较低。

1992 年 7 月第一个 ALSP 系统正式投入使用，应用于美军的“中心堡垒 92”、“回师德国 92”和“聚焦镜头 92”3 次主要军事演习。

ALSP 应用范围远没有 DIS 2.X 广泛，原因在于其强烈的军事背景，针对离散事件和逻辑时间的仿真，难以与实时、连续、平台级的实体互操作。ALSP 与 DIS 2.X 融合比较困难，因为聚合成的部队级实体需要解聚成为单独的实体，在实时仿真时钟下实现 DIS 2.X 系统中实体的交互，完成交互后在非实时的条件下完成重新聚合成部队级的实体。这些，目前 ALSP 的体系结构和技术条件难以实现。

11.3.2 分布交互式仿真基本原理

1. 分布交互式仿真系统的特点

分布交互式仿真系统从结构上可以视为由仿真节点和计算机网络组成。仿真节点本身可以是一台独立的仿真计算机，甚至是一个仿真系统。与单独仿真计算机不同，DIS 的仿真节点不仅要完成本节点的仿真任务(如运动学、动力学模型的计算，人机交互，仿真图形或动画的生成等)，还要负责将本节点的有关信息发送到其他节点；同时，它也必须按一定周期接收其他节点发送来的信息，并作为执行本节点任务时的输入或条件。

DIS 除了在功能上具有分布性以外，在地域上往往也是异地分布的。根据其地理分布的广度，计算机网络也可采用多种形式。如果相对比较集中，则可采用局域网(local area network，LAN)，如果分布比较分散，则需采用广域网(wide area network，WAN)或通过网桥(bridge)、路由器(router)、网关(gateway)等互连设备构成网络。

一种典型的 DIS 结构如图 11.13 所示。

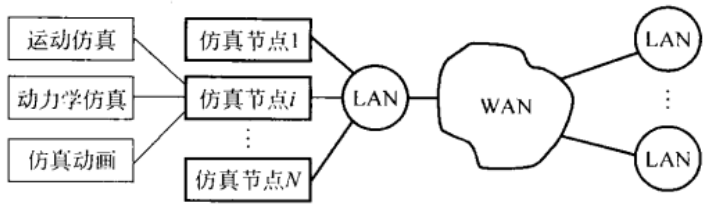


图 11.13 典型的 DIS 结构

我们从“分布交互式仿真系统”的名称就不难分析出此类并行仿真系统的特点，即分布性、交互性。

如前所述，分布性包含功能及地理位置两方面的分布。地理位置上的分布性通过网络实现其集成，而功能分布性表现在：在 DIS 中没有中央处理机，各仿真节点是平等的。每个仿真节点具有各自的资源，并负责对分配给它的仿真任务进行处理。DIS 中节点一般具有自治性，即是说，一个节点的加入与退出不会引起别的节点正常执行。实际上，大多数 DIS 不是执行单一仿真任务的系统，而往往是将功能各异的若干独立仿真器集成起来以完成更大规模的系统仿真。

DIS 的交互性包括三方面的内容：

① 节点内部进程交互、人机交互；

② 节点之间的交互作用；

③ 不仅人机交互及节点之间的交互要求实时性，而且要保证空间的一致性，我们称其为时空一致性。时空一致性是 DIS 的基本要求，也是 DIS 的本质特征。

一般说来，DIS 中的节点往往是任务各异的，这要根据仿真系统的任务分配而定；而且，也并不要求计算机的类型、操作系统类型、编程语言类型一定相同。为了保证这种异构系统的信息共享与通信，需要在应用层定义信息交换的格式和内容。这些交换的信息单元称为协议数据单元(protocol data units，PDU)，使得交换的信息内容与通信硬件和软件无关，从而，易于采用通用的商业网络软硬件环境(如 TCP/IP 协议)。

2. DIS 的时钟同步

时钟同步的目的就是要在 DIS 的各节点间建立一个全局统一的时间基准。由于每一个节点均有其独立的计算机系统时钟和仿真时钟，为保证 DIS 的时间统一性，除了要将仿真时钟设置相同的初始值，并让仿真时钟保持与计算机系统时钟同步外，还必须考虑各计算机时钟的漂移带来的误差，因此，还要求各节点计算机系统时钟保持同步。

保持 DIS 时钟同步最准确的办法是采用全球定位系统(GPS)接收机的高分辨率的时钟作为基准时钟。由于 GPS 的原子钟群的漂移率约为 10^{-13}s/s，即 30 万年的累计误差为 1s，这种误差对 DIS 来说完全可以不必考虑。这种方法的缺点是费用太高。实际上，现代计算机基于石英振荡电路的时钟系统的漂移率大多也可达到 10^{-6}s/s，即一个小时的累计误差大约为 3.6ms，这意味着两个节点的系统时钟在初始值相同的情况下，经过 1 小时的累计误差不大于 7.2ms。如果以某一节点的系统时钟为基准设定初值，只要有限的几次校正就可保证数小时仿真计算的同步。

问题的关键是节点间仿真时钟的同步。当一个节点通过网络向其他节点发送信息时，由于网络传输时间的不确定性，接收节点无法确知这些信息是何时产生的。因此，所发送的信息不仅要包括仿真对象的状态数据(位置、速度、加速度等)，还要包括产生这些数据的时间，称为时间戳(stamp)。节点之间通过时间戳进行同步是 DIS 时钟同步的重要特征。

3. DR 算法

在仿真过程中，DIS 的各节点之间交互性表现在实时地交换信息。采用时间戳机制保证了节点之间的时钟同步，但并未解决信息的实时性。实际上，由于 DIS 的规模一般比较大，网络上的节点较多，每个节点所包含的实体也很多，这样，在网络需要交换大量的信息量，而且交换频度也很高。然而，由于网络的通信速度的限制，任何信息的交换必然带来延迟，而且延迟的时间往往带有随机性。这些给 DIS 的并行仿真带来巨大困难。

如何减少信息的交换量和交换频度，而又能保证恰当的仿真精度是 DIS 需着重解决的问题。目前，DIS 普遍采用 DR(dead reckoning)算法。dead reckoning 一词来源于航海。在航海中，为了使船只到达某一目标地，一般采用将船头保持某一方向航行一段时间，只有当实际航线与期望方向出现较大差异时，再调整航行方向。

将这一思想用于 DIS 时，其做法是：相关节点不但保存本节点仿真对象的模型，而且也保存对方的 DR 模型，还保存自身的 DR 模型。这样，任意两节点之间不必按仿真步频繁交换数据，而是利用 DR 模型来推算相关节点的状态(参见图 11.14)。

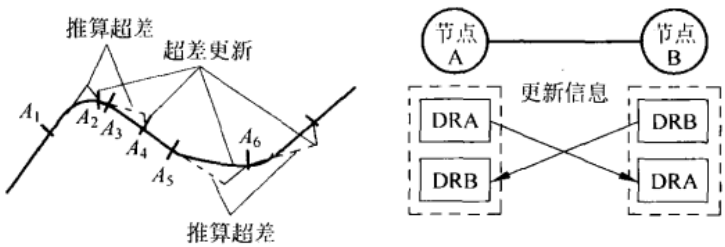


图 11.14 DR 原理图

例如，考虑两节点 A、B，如果节点 A 仿真红方的战斗机 F_A，节点 B 仿真蓝方的战斗机 F_B。显然，双方的飞行轨迹既具有独立性，又具有相关性。设在初始点 A_1，B_1 两节点是同步的，随着仿真往前推进，A 节点不仅计算仿真模型，同时进行 DRA 模型计算，并将两者的结果进行比较，称为“推算超差”，当超差超过一定的阈值 δ_A 时(譬如图中的 A_2 点)，就向节点 B 发送当前 F_A 的状态值及时间戳。这时，节点 B 接收到信号后，就要调整位于该节点的 DRA 模型的状态值。同样的原理，F_B 所在节点 B 也进行自身的仿真模型及 DRB 模型计算，当超差超过其阈值 δ_B 时才向 A 节点发送当前 F_B 的状态值。这样，就可大大减少交换数据的次数，并保持一定的精度。

对一个节点来说，除了进行本节点的对象仿真模型计算之外，还需计算本节点对象的 DR 模型及相关节点对象的 DR 模型，从而将增加计算工作量。所以，DR 模型不能太复杂而又能保持一定的精度。一般说来，DR 模型主要用于空间状态的计算，下面讨论用于空间状态的常用 DR 模型。

实体空间状态可用位置及姿态来描述。实体位置通过空间坐标、速度、加速度定义，姿

态则通过其姿态角及转动角速度来定义。为了便于讨论，现将这些变量说明如下：

x, y, z——仿真实体相对地面坐标系的空间位置，初值为 x_0, y_0, z_0；

v_x, v_y, v_z——仿真实体在地面坐标系三个方向的运动速度，初值为 v_{x0}, v_{y0}, v_{z0}；

$\ddot{a}_x, a_y, a_z$——仿真实体在地面坐标系三个方向的运动加速度，初值为 a_{x0}, a_{y0}, a_{z0}；

ψ, θ, φ——仿真实体相对地面坐标系的空间姿态角(欧拉角)，初值为 $\psi_0, \theta_0, \varphi_0$；

$\omega_x, \omega_y, \omega_z$——仿真实体在机体坐标系的转动角速度，初值为 $\omega_{x0}, \omega_{y0}, \omega_{z0}$。

常用空间状态 DR 算法有：

① 一阶外推算法，空间位置的推算公式为

$$\begin{cases} x_k = x_0 + v_{x0} \cdot k \cdot \Delta T \\ y_k = y_0 + v_{y0} \cdot k \cdot \Delta T \\ z_k = z_0 + v_{z0} \cdot k \cdot \Delta T \end{cases} \tag{11.8}$$

其中，ΔT 为推算步长(一般取仿真步长)，k 为推算步数。为了推算姿态角，需要根据 ω_x，ω_y, ω_z 计算出 $\dot{\psi}, \dot{\theta}, \dot{\varphi}$，为此，进行坐标转换

$$\begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\psi\tan\theta & \sin\psi\tan\theta & 1 \\ -\sin\psi & \cos\psi & 0 \\ \cos\psi\sec\theta & \sin\psi\sec\theta & 0 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{11.9}$$

从而，三个姿态角的一阶外推式为

$$\begin{cases} \psi_k = \psi_{k-1} + \dot{\psi}_{k-1}\Delta T \\ \theta_k = \theta_{k-1} + \dot{\theta}_{k-1}\Delta T \\ \varphi_k = \varphi_{k-1} + \dot{\varphi}_{k-1}\Delta T \end{cases} \tag{11.10}$$

② 空间位置的二阶 DR 外推算法

空间姿态一般采用一阶算法，但位置有时采用二阶外推算法，其表达式为

$$\begin{cases} x_k = x_0 + v_{x0} \cdot k \cdot \Delta T + \dfrac{1}{2} a_{x0} \cdot (k \cdot \Delta T)^2 \\ y_k = y_0 + v_{y0} \cdot k \cdot \Delta T + \dfrac{1}{2} a_{y0} \cdot (k \cdot \Delta T)^2 \\ z_k = z_0 + v_{z0} \cdot k \cdot \Delta T + \dfrac{1}{2} a_{z0} \cdot (k \cdot \Delta T)^2 \end{cases} \tag{11.11}$$

当状态更新信息到达接收节点时，接收节点要对 DR 模型进行更新。DIS 的传输总会存在延时，即接收到信息的时间较实际发送的时间要滞后。因此，接收节点在更新 DR 模型时，要根据发送时间戳来推算当前时刻的状态值，这称为延时补偿。显然，根据发送时间戳推算得到的当前时刻的状态值较原来的 DR 模型推算值会有差别，如直接刷新，则必然出现画面抖动，从而影响仿真的效果。为此要对这种可能出现的抖动进行平滑处理。

平滑公式的一般形式为

$$\hat{x}_i = x_{\text{old}} + i\frac{x_{\text{new}} - x_{\text{old}}}{n} \tag{11.12}$$

其中 $\hat{x}_i$ 为第 i 次的平滑值，x_{old} 为更新前 DR 模型推算得到的当前时刻的状态值，x_{new} 为进行延时补偿后的新值，n 为平滑的次数。实质上，这是通过 n 次处理来达到按新值外推的 DR

模型所得到的值。一旦平滑结束，就转入到 DR 外推计算。需要说明的是，延时补偿及平滑处理仅针对接收节点，而对发送节点来说，则不必进行此类操作。

从并行仿真的角度来看，空间的分布性及系统的交互性是 DIS 结构的主要特征。DIS 的关键是如何保证各节点的时空一致性。由于网络传输速度的限制，DR 算法成为解决时空一致性的核心。

11.4 基于高层体系结构的仿真

SIMNET、DIS、ALSP 都是同类功能仿真应用(武器平台、模拟仿真器、计算机生成兵力、聚合级仿真模型)互联，只有有限的互操作性，不能满足越来越复杂的作战仿真需求。为此，美国国防部 1995 年发布的建模与仿真主计划，决定在吸收和继承 DIS 与 ALSP 协议有益思想的基础上，在国防部范围内建立一个通用的仿真技术框架来保证国防部范围内的各种仿真应用之间的互操作性。技术框架的核心是高层体系结构(HLA)。HLA 在 1996 年 8 月完成基础定义，随后为北约各国采纳，并于 2000 年 9 月被 IEEE 接受为标准 P1516.2。美国国防部规定 2001 年后所有国防部门的仿真必须与 HLA 相容。HLA 的提出，标志 DIS 技术发展到成熟阶段。

11.4.1 HLA 体系结构

HLA 是一种软件体系结构而不是一种实现。在进一步理解 HLA 之前，先列出 HLA 中的重要术语如下：

① 联邦(federation)：由仿真子系统联合构成的仿真系统。

② 联邦成员(federate)：构成联邦的每一个仿真子系统。

③ 联邦执行(federation execution)：一个联邦运行的整个时期。

④ 运行支撑环境(runtime infrastructure，RTI)：HLA 的支撑软件。

⑤ 联邦对象模型(federation object model，FOM)：联邦中联邦成员进行数据交换的公共对象模型。

⑥ 对象模型模板(object model template，OMT)：规定每一个 FOM 可以采用的结构，OMT 是所有 FOM 的元模型。

⑦ 仿真对象模型(simulation object model，SOM)：遵循 OMT 的联邦成员模型。

体系结构按照 Shaw 和 Garlan 的定义，软件体系结构理论上是包括建立系统的元素描述、元素间的交互、指导元素合成的模式和对这些模式的限制。HLA 的主要目标是允许通过联合其他仿真来实现仿真的应用，因此支持基于组件的仿真开发。HLA 是一种支持基于组件的仿真体系结构。组件可以理解为联邦成员，而联邦成员是一个仿真系统或工具。联邦由联邦成员构成，联邦成员是构建联邦的基本成员，也是软件重用的基本单元。因此联邦成员是完整的运行子系统，而不是库里的例程或对象。

作为一种体系结构，HLA 标准定义元素、交互、模式如下：

① 元素：包括联邦成员、RTI 和公共对象模型，由规则和接口规范定义。

② 交互：对于一个给定的联邦，联邦对象模型定义了联邦成员和 RTI 间交互所传递的数据，从而定义了联邦成员间（总是通过 RTI）的交互，由规则和接口规范定义。

③ 模式：HLA 中允许合成模式由规则进行限制，并在接口规范中加以说明。

因此，HLA 标准的定义包括 3 个部分：

① 框架与规则集（framework and rules）IEEE P1516；

② 联邦成员接口规范（federate interface specification）IEEE P1516.1；

③ 对象模型模板（object model template）IEEE P1516.2。

三者的关系是：规则集中规定了联邦对象模型（FOM）和仿真对象模型（SOM）的建立必须用 OMT 记录，OMT 保证联邦成员与 RTI 进行方便的交互，三者相互作用，紧密相关，任何一部分的改动都会影响其他部分的改变。

HLA 的核心是 RTI，它将应用层同其底层支撑环境功能分离，即将具体的仿真功能实现、仿真运行管理和底层传输三者分离开来，从而使各部分的开发与实现相对独立。RTI 提供 6 大类服务，即联邦管理、声明管理、对象管理、所有权管理、时间管理和数据分发管理。从联邦成员看 RTI，联邦成员的变化不会影响 RTI，不同的联邦成员可使用同一个 RTI，只要它们遵循相关规则和规范。从 RTI 看联邦成员，RTI 为联邦成员提供接口，尽管联邦成员的所有状态被屏蔽，同时联邦成员也为 RTI 提供接口。RTI 有多个联邦成员，每一个都有同样的接口，只是标识不同而已。RTI 提供给每一个联邦成员的接口称为 RTI 大使（RTIambassador），联邦成员调用该接口的操作请求 RTI 服务，每一个联邦成员也对 RTI 提供一个称为联邦成员大使（federate ambassador）的接口，当 RTI 必须通知联邦成员时，RTI 通过调用联邦成员大使的操作来实现。RTI 与联邦成员之间的接口示意如图 11.15 所示。

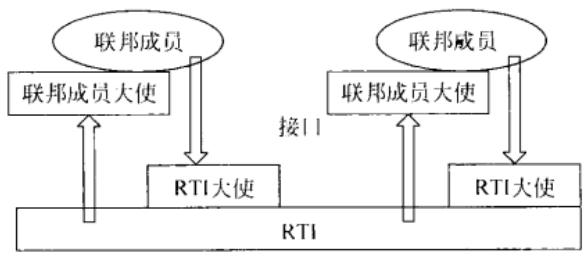


图 11.15 RTI 和联邦成员之间的接口

HLA 是一个层次化的体系结构，每一层都对其上一层提供服务，并接受下一层服务。从联邦成员来看，RTI 表现为一个层次，在 RTI 下面完整地封装了 RTI 的功能。

HLA 是一个数据抽象的体系结构。HLA 是面向编程的，数据表示和它的相关原始操作被封装在一个抽象的数据类型或对象中。根据 HLA 的规则，联邦成员之间永远不会直接发生交互，总是通过联邦的 RTI 进行通信与交互，联邦成员间不需要保持任何种类的调用，实际上甚至不需要知道其他联邦成员的存在。

HLA 还是一个基于事件的体系结构。基于事件的体系结构也称为隐含调用、反应集成或选择性广播。隐含调用相对于直接调用。一个组件可以宣布或广播一个或多个事件，系统中其他组件通过一个过程与事件联系起来用于登记对某个或几个事件感兴趣。当事件

被宣布，系统自动调用所有登记过该事件的过程，从而引发其他组件的调用。

与 DIS、ALSP 比较，HLA 体系结构更为合理，更为完善，它们三者之间的比较如下（参见表 11.2）。

表 11.2 DIS、ALSP 和 HLA 的综合比较

DIS	ALSP	HLA
平台级建模（以连续系统为主）	聚合级建模（以离散事件系统为主）	平台级和聚合级建模（包括连续系统和离散事件系统）
虚拟仿真/实况仿真	构造仿真	虚拟仿真/实况仿真/构造仿真
广播式通信/点对点通信	广播式通信	一点对多点通信
数据通信 PDU 规范（IEEE 1278—1）	规定联邦的数据交换，在接口控制文件中说明，但无标准格式	在 FOM 和 SOM 中定义数据通信格式
网络接口按 IEEE 1278—2 标准	由 AIS（ALSP infrastructure software）提供网络接口	由 RTI 提供网络接口
实时仿真	守恒的时间管理模式	多种时间管理服务

11.4.2 HLA 标准

HLA 标准包括三部分：规则、对象模型模板和接口规范，从而形成 HLA 标准的文档，如图 11.16 所示。本节分别对它们进行讨论。

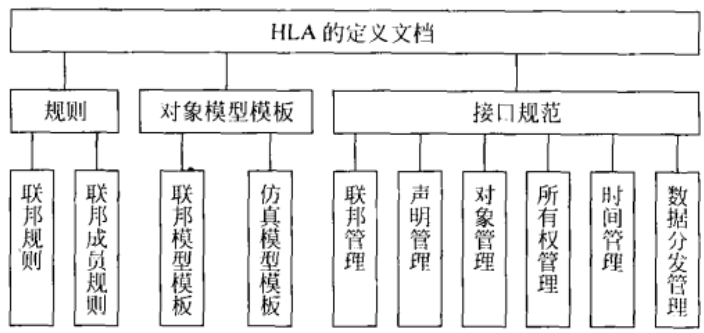


图 11.16 DMSO 定义的 HLA 标准文档

1. HLA 定义的规则

HLA 规则是构成 HLA 标准的 3 个重要的组成部分之一，它给出了 HLA 兼容的联邦成员和联邦的目标和限制，规定了联邦应用方式，其中前 5 条用于联邦，后 5 条用于联邦成员。

1）联邦规则

规则 1：联邦应该有一个联邦对象模型即 FOM，FOM 遵循 HLA 对象模型模板即 OMT。

FOM 是一个联邦的公共词汇表，它描述联邦成员向联邦中其他成员展示的对象和交互。FOM 定义是建立联邦的重要步骤。OMT 需要的信息比 RTI 执行时所需要的要多，其

中模板中附加的信息是为了确保联邦中语义的一致性。

规则 2：在联邦中，所有与仿真有关的对象实例的描述应该在联邦成员中，而不是在 RTI 中。

本条规则表明，RTI 服务对仿真互操作是通用的服务，RTI 不用修改可应用于不同的仿真应用领域，例如，同一个 RTI 可以为飞机驾驶模拟服务，也可为制造调度仿真、道路交通仿真服务。

本条规则也对 RTI 的实现进行了限制，RTI 永远不应成为属性数据库，即 RTI 不保存与其服务相关的状态，当需要实例的属性值或参数时，它总依赖联邦成员来提供。

规则 3：在联邦执行过程中，联邦成员间所有 FOM 数据的交换都应通过 RTI 来实现。

这条规则保证了仿真组件能互操作和可重用，RTI 不能被联邦成员绕过，这样，所有联邦成员的交换都是可重用的。

规则 4：在联邦执行过程中，联邦成员和 RTI 之间将遵循 HLA 接口规范进行交互。

这条规则确认了接口规范在 HLA 中的地位。不仅联邦成员的交互只能通过 RTI 完成，而且它们的交互还要遵循 HLA 接口规范。这使联邦成员与不同的 RTI 的特殊性隔离开，使得一个 RTI 的实现能被另一个 RTI 替换。

规则 5：在联邦执行的任何时刻，一个实例属性最多只能由一个联邦成员拥有。

这就意味着，一个实例属性，谁拥有，谁负责更新。后面将会看到，接口规范规定，如果联邦成员不拥有某个实例属性，则其更新该实例属性的任何尝试均会被 RTI 拒绝。

2）联邦成员规则

规则 6：联邦成员应该有一个仿真对象模型即 SOM，SOM 遵循 HLA 的对象模型模板即 OMT。

对于联邦成员，它向外部展示的数据或仿真功能必须通过 SOM 来定义，OMT 建立了表述联邦成员特征所需要的最小信息集。

规则 7：联邦成员应该能更新和/或反射其 SOM 中规定的任何属性，发送和/或接收其 SOM 中规定的交互。

本条规则定义了 HLA 的数据兼容性：SOM 中定义的属性和交互必须完全被 RTI 支持。联邦成员会启动与 RTI 有关的行为，且响应 RTI 对其 SOM 中的属性和交互所启动的服务。

规则 8：在联邦执行过程中，联邦成员应能按 SOM 中的规定，动态地转移和/或接收属性的所有权。

联邦成员不仅要支持 SOM 中数据的产生和消费，而且要支持联邦成员实现部分接口规范定义的属性所有权转移协议。联邦成员是否愿意转移所有权必须在其 SOM 中声明。

规则 9：联邦成员应该能按 SOM 中的规定，改变更新属性的条件（例如阈值）。

SOM 定义联邦成员更新给定属性的条件（时间段、阈值界限等），联邦成员应具有改变这些条件的能力。

规则 10：联邦成员应能管理局部时间，从而允许与联邦中其他成员协调数据交换。

联邦成员局部时间管理可以利用 RTI 的时间管理函数，RTI 支持多种时间管理策略，并允许联邦成员具有不同的时间管理策略。当然联邦成员也可以没有时间管理服务，这意味着不能基于时间来与其他成员协调数据交换。

2. HLA 信息模型——对象模型模板(OMT)

OMT 规定了所有 FOM(联邦模型模板)的结构,每一个联邦均有对应的 FOM。FOM 中只定义联邦成员间共享信息,不描述联邦成员内部的事情。FOM 是联邦运行时通过 RTI 交换数据的唯一词汇表。在联邦运行的开始,FOM 作为数据提供给 RTI。当 FOM 改变时 RTI 并不需要改变,这是 HLA 的一个重要特征,这意味着模型数据驱动仿真试验,而试验环境 RTI 可以适应不同的仿真模型,只要通过 OMT 定义的 FOM 即可。

OMT 的主要组件包括交互类与对象类,下面分别加以说明。

1) 交互类

交互类是在一个时刻通过 RTI 传送到其他联邦成员的数据集,用于表示仿真模型中有多个联邦成员感兴趣的事件。交互在仿真时间点上发生。

一个联邦成员发送一个交互,其他联邦接收这个交互。每一次交互传送一组数据,接收后交互不再继续存在。

FOM 定义了交互类。一个联邦成员发送的一个交互是一个特定类的交互。交互类是一个单向继承的层次结构,层次的根叫交互根(interaction root)。每一个交互类定义由它发起的参数,每一个子类继承其父类定义的参数。默认情况下,交互根不定义参数。一个类的完整有效的名字是由根到类的名字组合而成,中间用"."分隔。在 FOM 中,完整有效的类名必须是唯一的。采用 UML 来描述 FOM 交互类如图 11.17 所示。

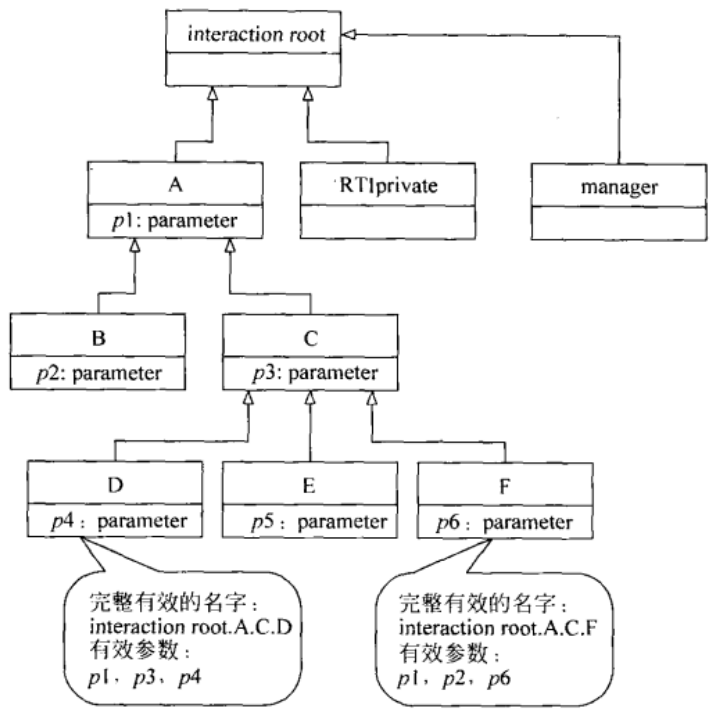


图 11.17 采用 UML 来描述 FOM 交互类层次结构

2) 对象类

OMT 定义的对象类,每个类有一个名字,每个类定义一个称为属性的数据集(可能为空)。联邦成员产生这些类的实例。联邦中,每个实例都有自己的标识,都有独特的实例属

性。仿真时，联邦成员负责更新对象实例的状态。

RTI 中的对象是指：仿真实体和多个联邦成员有关并因此由 RTI 来处理，在仿真期的某些时段持续存在。由于联邦成员只有通过 RTI 才能实现相互对话，因此每个联邦成员必须把仿真实体内部的概念转换为 FOM 中定义的 HLA 对象。

对象类也是层次结构。每个对象类一定有一个直接的父类，对象类形成单向继承的树。根类称为对象根(object root)。与交互类一样，一个类的完整有效的名字是由根到类的名字组合而成，中间用“.”分隔。在 FOM 中，完整有效的类名必须是唯一的。

每一个类继承其父类的所有属性。一个类声明的属性连同其从父类继承来的属性统称为有效属性。根类有一个必需的属性，即删除对象的特权(privelege to delete object)，因此，每个对象类至少有一个有效属性。采用 UML 来描述 FOM 对象类如图 11.18 所示。

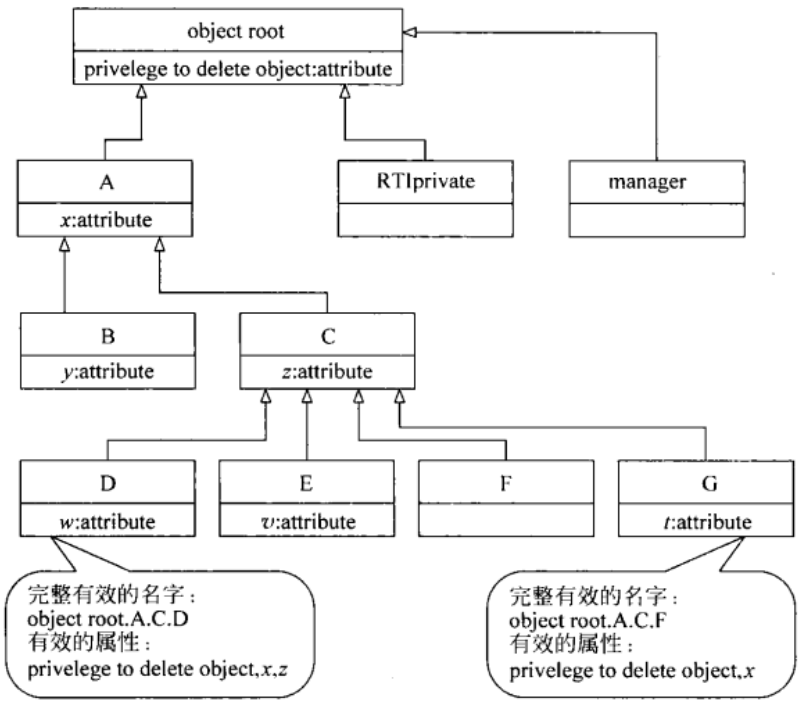


图 11.18 采用 UML 来描述 FOM 对象类层次结构

那么到底是用交互类还是用对象类建模呢？从某种意义上讲，两者是可以转换的。一般来说，任何发生的事情或事件用交互类建模，而如果建模的实体有持续状态，这个实体应该表示为对象。

3. HLA 接口规范

接口规范定义了 RTI 为联邦成员提供服务与联邦成员为 RTI 提供服务的接口要求。DoD DMSO 于 1998 年 4 月 20 日推出 1.3 版，这些服务根据其用途的相似性划分为 6 组，下面分别加以说明。

1) 联邦管理

联邦管理服务主要用于创建、动态控制、修改和撤销联邦执行。联邦执行，就是在联邦

运行过程中，通过 RTI 与成员之间的互操作构造的一个虚拟世界，可以将其理解为具有一定生命周期的活动联邦。

联邦管理服务共 20 项，可分为 4 小组。第一小组用于完成联邦执行的创建、撤销以及联邦成员的加入与退出。联邦执行过程始于联邦执行建立，联邦建立后，联邦成员可以随时加入与退出。当所有成员退出后，联邦执行才能被撤销。第二小组用于联邦成员间的同步，第三小组用于联邦保存，第四小组用于联邦恢复。

2）声明管理

在 DIS 协议中，仿真系统的交互是通过互相发送 PDU 来完成的，而 PDU 的发送采用广播方式，因此当仿真节点数 N 增加时，网络的通信量将以 N^2 数量级增长，而且每个仿真节点系统在收到一个 PDU 后，都要判断该 PDU 是不是发给自己的，这个过程浪费了大量的处理时间，限制了构建一个大型分布式仿真系统的能力。

为解决 DIS 协议中的问题，HLA 采用了一种匹配机制，即数据生产者向 RTI 声明自己能生产的数据，数据消费者向 RTI 订购自己需要的数据，由 RTI 负责在生产者和消费者之间进行匹配，这种匹配可以在类（对象类或交互类）层次上进行，也可在实例（属性实例或交互实例）层次上进行。

声明管理（declaration management）的目的是在联邦范围内建立一种发布和订阅关系，为联邦成员提供类层次上的表达（发布或订购）机制，以利用 RTI 机制进行联邦成员间的交互。

联邦成员发布对象类，意味着它能对该类的对象建立动态模型，更新对象的状态。成员订阅对象类，表示它希望接收该交互类。

HLA 数据交换采用隐含调用，联邦成员不是通过名字进行交互，而是把数据先传送给联邦，由 RTI 确保发送到对数据感兴趣的联邦成员。RTI 使用这些声明来安排数据的路由、转换数据以及管理兴趣。路由是指数据产生或更新时 RTI 使用订阅来决定该数据通知哪个联邦成员。转换数据用于当 RTI 收到数据后，要根据联邦成员的订阅完成修剪和推送（包括贴标签），从而保护联邦成员免受 FOM 扩展的影响。管理兴趣，则用于告知发布者，它所发布的数据是否有成员感兴趣，即是否有成员订阅，从而当其他联邦成员不需要时，发布者停止产生数据，从而实时地提高系统运行效能。

声明管理服务分为 3 个小组，第一小组用于发布管理，第二小组用于订阅管理，第三小组用于兴趣管理。

3）对象管理

对象管理（object management）是在声明管理的基础上，实现对象实例的注册/发现、属性值的更新/反射、交互实例的发送/接收，以及对象实例的删除等，共有 17 项。

4）所有权管理

所有权管理（ownership management）是 HLA 接口规范的重要内容之一。在联邦执行的任何时刻，对象实例的任一属性应该有一个且只有一个联邦成员来负责其值的更新，这时称该成员拥有该属性的所有权。属性的所有权可在成员间转移，所有权管理服务用于处理属性所有权的迁移与接受。这类服务共有 16 项。

所有权转移的模式有两种：一种是“推”式，由希望放弃实例属性所有权的联邦成员向 RTI 发出请求转让申请，然后在 RTI 协调下完成转移；另一种是“拉”式，由希望得到实例

属性所有权的联邦成员向 RTI 发出申请，然后在 RTI 协调下完成转移。

5）时间管理

联邦成员在它们各自的线程控制下执行，因此，保证成员间的正确顺序是至关重要的。在 HLA 中，事件基于“仿真时间”排序。RTI 的时间管理服务要实现：

（1）允许每一个联邦成员与其他联邦成员协调，以推进自己的仿真时间；

（2）控制时间戳事件的发送，保证联邦成员不会接收来自其他联邦成员的“过去”事件。

HLA 提供了 23 项时间管理(time management)服务。多年来，人们对这一部分服务进行了众多研究，并根据应用进行了扩展。

6）数据分发管理

数据分发管理(data distribution management，DDM)服务控制着联邦成员间数据生产者与消费者的关系。HLA 声明管理在属性层次上为联邦成员提供了表达发送和接收意图的机制，而 DDM 根据对象实例和路由空间进行管理，其目的是减少仿真运行过程中无效数据的发送与接收，从而提高运行效率。

支持数据分发的基本概念是路由空间(routing space)。它是一个多维关联空间，联邦成员可用它来描述其希望接收和/或发送的数据范围。路由空间的子集称为区域(region)。联邦成员可以指定订阅区域，这就意味着通知 RTI，只将该区域的数据发送给它。

HLA 提供 12 项数据分发服务。需要说明的是，所有数据分发管理服务都独立于时间管理，即服务一旦执行，就立即生效，不受时间管理的限制。多年来，人们对这一部分服务也进行了众多研究，并根据应用进行了扩展。

11.4.3 HLA 运行支撑环境(RTI)

运行支撑环境(run-time infrastructure，RTI)是 HLA 仿真系统进行分层管理控制、实现分布仿真可扩充性的支撑基础。运行支撑环境(RTI)是按照 HLA 开发的服务程序，它实现了 HLA 接口规范中所有的服务功能。

如果说 HLA 是建模与仿真的技术框架的核心，那么 RTI 则是核心中的核心，建立联邦成员必须以 RTI 为其支撑软件，仿真过程中需要交互的各种信息都要通过 RTI 来完成。

1. RTI 的特点与作用

从 2000 年 HLA 成为 IEEE 的国际标准 IEEE 1516 以来，各种不同版本的支持 HLA 的 RTI 已经开发出来。国外有美国国防部 DMSO RTI，瑞典 pitch 公司 pRTI，美国 MAK 公司 MÄK RTI。国内有国防科技大学机电学院 KD—RTI，国防科技大学计算机学院 StarLink，北京航空航天大学自动化学院 AST—RTI，北京航空航天大学计算机学院 BH HLA/RTI，航天工业总公司第二研究院 SSS—RTI 等。

RTI 具有可移植性和互操作性，它将仿真应用与数据传输分离开，在某种程度上实现软总线的功能，使联邦成员能够灵活地加入仿真执行。实际上，HLA/RTI 可以看作 ISO 中的应用层协议，它提供了独立于系统的服务来协调网络间应用程序(联邦成员)的通信，这样使得仿真应用程序开发者只需要集中于仿真功能的开发，不必再涉及有关网络通信和仿真管理等方面的实现细节。RTI 的具体作用表现在以下几个方面：

(1) 它具体实现了 HLA 接口规范。为了实现联邦内部各个联邦成员之间进行高效的信息交换，HLA 接口规范用文字定义了各种标准服务和接口，而 RTI 则用程序设计语言将这些标准的服务和接口转换成标准的 RTI API 函数，使得基于 HLA 的仿真开发成为可能。

(2) 它为仿真应用提供了仿真运行管理功能，比如仿真过程的开始、暂停、恢复、时间同步、仿真时钟推进等。开发人员只需集中精力实现具体的仿真功能，仿真运行管理则由 RTI 来完成，不像过去传统的仿真程序设计一样，开发人员不仅要完成具体的仿真功能实现，而且还要实现复杂的仿真运行管理功能。

(3) 它提供了底层通信传输服务，屏蔽了网络通信程序实现的复杂性。开发人员可以很容易地实现数据的发送和接收，从而降低分布交互仿真程序设计的复杂程度，而且这种传输机制允许各个联邦成员进行不同级别的数据过滤，极大地减少网络数据流量，提高仿真系统的运行效率。

(4) 它是仿真功能、仿真运行管理和底层通信传输服务三者分离的基础。RTI 使仿真系统具有较好的扩充性，便于实现仿真系统中各个组成部分的“即插即用”，因此各个组成部分的编程实现可以相对独立地进行，很适合团体开发。

2. RTI 的组成部件及组件之间的关系

这里，我们以从 DMSO 下载的 RTI1.3v6 为例，介绍 RTI 软件的构成以及组件之间的关系。

RTI1.3v6 的组成部件由 RTI 执行进程(RTIExec)、联邦执行进程(FedExec)以及 LibRTI 库组成。其中 RTIExec 负责管理联邦执行的创建与执行，每一个联邦执行都由一独立的 FedExec 进行标识；FedExec 管理该联邦中各成员的加入和退出；LibRTI 库将 RTI 的服务扩展给成员的开发人员，该成员与其他成员的信息交互就是通过调用它的成员函数来实现的(参见图 11.19)。

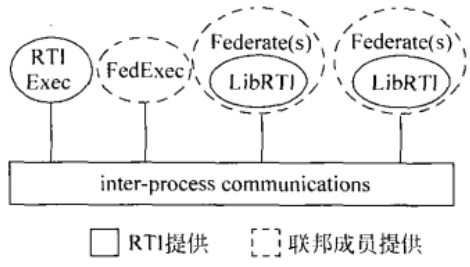


图 11.19 RTI 组成

从图 11.19 中可以看出，在一个联邦成员当中包含两部分的内容：第一就是联邦成员自身的代码；第二是 LibRTI 部分。前者实现成员的仿真模型，后者则用来完成成员的信息交互。

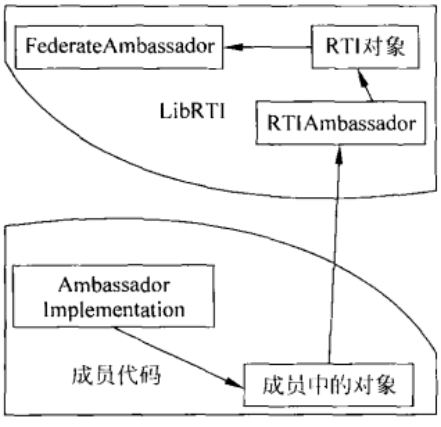


图 11.20 成员代码和 LibRTI 的关系

在图 11.20 中，给出了它们两者之间的关系。其中成员代码部分包含有模型所需要的各种对象的实现以及成员代理的实现；LibRTI 部分包含 RTIAmbassador 和 FederateAmbassador 两个代理类以及 RTI 对象，成员就是通过使用 LibRTI 来完成与 RTIExec，对应 FedExec 以及其他成员的通信的。图中的箭头表示一种调用或依赖关系。在 HLA 的接口规范中定义了 LibRTI 需要提供给各成员的服务，同时，它也规定了各成员对联邦应尽的义务，RTIAmbassador 和 FederateAmbassador 就是分

别针对这两种要求的软件实现，成员对 RTI 所发出的一切请求都是通过 RTIAmbassador 来实现的，而成员本身需要的外界信息则需要通过继承 FederateAmbassador 类来完成，所有的这些功能都在这个类的成员函数中实现。

这里需要指出的是成员代理类是一抽象类，每一成员的成员代理类在实现时都是继承了这一抽象类，在成员加入联邦时，成员代理类的对象实例作为一参数进行注册，这样在仿真过程中，当联邦接收到其他成员的请求（如发送信息）时，成员代理类中的回调函数就向成员提供获取这些信息的一种回调机制。

3. 使用 RTI 建立联邦成员

开发一个联邦成员主要包括两个方面的工作：第一是仿真模型的开发；第二是成员代理类的设计。这里重点对仿真模型的开发流程进行讨论。

1）基于 RTI 的成员的仿真流程

成员的仿真流程主要包括以下几个步骤：

（1）创建联邦

联邦是由第一个加入联邦的成员在调用 RTI 代理的联邦创建函数时所创建的，它给出了该联邦的名称以及 fed 文件，在这一文件中定义了各成员间需要交互的属性类和交互类，它由 OMDT 开发工具生成。

（2）加入联邦

成员调用 RTI 代理的成员加入函数来实现。在该函数中，必须指定要加入联邦的名称、成员名称以及这一成员的成员代理类的对象指针，函数返回该成员对应的句柄。

（3）信息的公布与订购

公布该成员需要向联邦中其他成员提供的信息以及订购仿真过程中所需要的其他成员的信息。

（4）设置时间推进模式

根据系统的需求设置该成员的时间推进方式及相关参数。RTI1.3v6 成员的时间管理策略有 regulating、constrained、regulating and constrained 以及 neither regulating nor constrained 四种。

（5）仿真模型的计算。成员的主要工作在这里完成。

（6）RTI 代理请求时间推进。

（7）调用 RTI 代理的 tick()函数。

由于 RTI1.3v6 不是多线程应用，本地 RTI 部件（LRC）需要一定的时间来处理诸如信息的交换等工作，tick()函数就是用来完成这一任务的，在其工作期间，成员代理对象的一些函数就可能被调用以获取新接收到的交互信息。

（8）如果仿真未结束，则返回到仿真模型计算，否则继续。

（9）成员退出联邦。

（10）删除联邦。由最后退出联邦的成员执行，如果联邦中还有未退出的成员，则会给出提示，并继续保留该联邦。

2）代码设计

根据上面给出的仿真流程，我们在这里以 C++语言为例，给出成员流程实现的部分

代码。

```
//定义 RTI 代理类和成员代理类的对象
RTI: : RTIAmbassador rtiAmb;
MyFederateAmbassador fedAmb;
//1)创建联邦
rtiAmb. createFederationExecution(fedExecName, fed-file name);
//2)成员加入联邦,返回成员句柄
federateHandle = rtiAmb. joinFederationExecution ( federate - name,
federation name, &fedAmb);
//3)公布、订购交互信息
RtiAmb. subscribeObjectClassAttributes ( obj - Handle,
AttributeHandleSet * param);
RtiAmb. publishObjectClass ( obj - Handle, AttributeHandleSet *
param);
RtiAmb. subscribeInteractionClass( Interaction-Handle);
RtiAmb. publishInteractionClass( Interaction-Handle);
//4)设置时间推进方式:既受限又约束
rtiAmb. enableTimeConstrained();
rtiAmb. enableTimeRegulation( grantTime, Lookahead);
//5)仿真模型的计算
依据不同系统的要求进行实现;
//6)请求时间推进
rtiAmb. timeAdvanceRequest( requestTime);
//7)调用 tick()函数
rtiAmb. tick(minTime,maxTime);
//8)判断仿真结束与否,若否,返回到//5);
//9)退出联邦
rtiAmb. resignFederationExecution();
//10)删除联邦。最后退出的成员完成
rtiAmb. destroyFederationExecution( fedExecName)。
```

11.4.4 小结

高层体系结构(HLA)定义了一个技术框架,是一个灵活的、可伸缩的、可重用的软件体系结构,通过它可创建基于组件的分布式仿真,构成系统的各类模块或各类仿真系统均可直接接入该框架,并能实现相互间的互操作及仿真部件的重用。

HLA 标准包括三个部分,即规则、对象模型模板和接口规范。HLA 协议的具体实现为 RTI,每个仿真节点通过逻辑通路与 RTI 相连,整个系统的交互通信由 RTI 进行协调和管理,各仿真节点间并无直接的联系。在 HLA 中,一个仿真节点只将它所仿真对象属性变化部分传输给 RTI,再由 RTI 将这些变化传输到需要它的仿真节点; HLA 允许每个仿真节点利用标准化的文档格式来确定自身产生何种信息,接受何种数据,以保证所有仿真节点对将要交换的数据达成一致,这样就打破了 PDU(协议数据单元)的限制,使不同类型的仿真有通信的可能,较好地解决了分布式仿真中的互操作、模型重用和伸缩性问题。

高层体系结构(HLA)是分布交互式仿真技术发展到成熟阶段的标志,它不仅已经被 DoD(美国国防部)应用于其建模与仿真中,也在民用领域得到了广泛的应用。

11.5 并行分布仿真的进展

HLA 是目前流行的分布仿真技术，自 2000 年被确立为 IEEE 1516 标准至今，其在世界军事仿真领域得到了广泛的应用。HLA 提出的最初目的是提高仿真组件的可重用性和互操作性，随着应用的不断深入，这一目标在一定程度上得到了实现。但是目前的建模与仿真系统依旧面临着一系列问题和挑战；例如，HLA 运行支撑环境(RTI)的实现与特定编程语言及计算机平台有关，互操作性有限，需要引入新的技术以实现仿真应用在更大规模上的互操作性；与其他领域的许多标准脱节，与诸多商业应用不兼容，特别是 Web 技术并未在建模与仿真领域得到广泛有效的应用等。为解决这些问题，建模与仿真需要确定和采纳有关的革新技术。

11.5.1 可扩展建模与仿真框架(XMSF)

目前，建立在 Internet 和 Web 技术之上的软件系统可以实现全球范围的互操作，一个基于 Web 的可扩展的框架能够为增强建模与仿真系统的功能提供可行的技术途径，从而满足训练、分析以及系统采办等方面的需求。通过采用 Web 技术作为一个共享的通信平台，目前的建模与仿真系统完全可以利用企业范围的软件开发模式及开发工具。鉴于此，美国海军研究生院(NPS)、乔治- 梅森大学(GMU)以及 SA IC 公司等组织的研究人员于 2002 年提出并启动了可扩展建模与仿真框架(extensible modeling and simulation framework，XMSF)。XMSF 代表了未来建模与仿真的发展方向，其核心是使用通用的技术、标准和开放的体系结构促进建模与仿真应用在更大范围的互操作性和重用性。

XMSF 使用各种开放标准(Web 服务、可扩展标记语言(XML)等)，定义了一组基于 Web 的建模/仿真标准、描述以及推荐准则等，为未来的建模与仿真应用创造了一个可扩展的框架。

Web 服务技术代表了分布式计算的下一个阶段的发展。可扩展建模与仿真框架定义为一组基于 Web 的建模与仿真的标准、描述(profiles)以及推荐准则的集合。基于 XML 的标记语言、Internet 技术与 Web 服务将促进新一代分布式建模与仿真应用的出现、发展与互操作。

XMSF 并不是单一的体系结构，而是诸多 profiles 的集合。仿真互操作标准组织(SISO)在 2003 年 9 月建立了 XMSF profiles 研究小组(XMSF ProfilesSG)，讨论确立了 XMSF profiles 的需求范围、定义、目标等。XMSF profiles 是描述基于可互操作 Web 应用的正式技术规范，能够促进建模与仿真的可组合性以及可重用性，有利于企业应用的集成。XMSF profiles 包括可应用的 Web 技术、协议规范、数据与元数据标准，经选择的现有标准的裁剪集合，以及关于应用实现的建议和指导等。XMSF profiles 的目标是：

① 提供对框架组件以及组件间接口的明确的功能说明；

② 确保在建模与仿真及其相关领域内已经存在的和新出现的 Web 使能(Web enabled)技术的互操作性；

③ 提供有利于促进组件在诸多建模与仿真应用领域可组合性及可重用性的必要的元数据；

④ 促进能够与现存的应用与服务互换的新应用与服务的发展；

⑤ 促进功能持续增强的新应用与服务的发展。

因此，XMSF 是为适应网络化要求对 HLA 的扩展和完善，并非取而代之。

11.5.2 面向服务框架(SOA)

面向服务框架(service oriented architecture，SOA)是指为了解决在 Internet 环境下业务集成的需要，通过连接能完成特定任务的独立功能实体实现的一种软件系统架构。在这一系统里每一个应用都被当做一个服务来调用和管理。

SOA 的关键是“服务”的概念，W3C 将服务定义为：“服务提供者完成一组工作，为服务使用者交付所需的最终结果。最终结果通常是使用者的状态发生变化，但也可能是提供者的状态发生变化，或者双方都发生变化。”

分布式计算将网络上分布的软件资源看作是各种服务。面向服务架构是一种不错的解决方案。但这种架构不是什么新思想；CORBA(通用对象请求代理体系结构)和 DCOM(分布式组件对象模型)就很类似，但是，这些过去的面向服务架构都受到一些难题的困扰：首先，它们是紧密耦合的，这就意味着分布计算连接的两端都必须遵循同样 API 的约束。如果一个 DCOM 对象的代码有了更改，那么访问该对象的代码也必须作出相应更改。其二，这些面向服务架构受到厂商的约束。Microsoft 控制 DCOM 自不必说，CORBA 也只是一个伪装的标准化，事实上，实现一个 CORBA 架构，经常都是在某个厂商对规范的实现上进行工作。

Web service 是在改进 DCOM 和 CORBA 缺点上的努力。今天应用 Web service 的面向服务架构与过去不同的特点就在于它们是基于标准以及松散耦合的。广泛接受的标准(如 XML 和 SOAP)提供了在各不同厂商解决方案之间的交互性。而松散耦合将分布计算中的参与者隔离开来，交互两边某一方的改动并不会影响到另一方。这两者的结合意味着公司可以实现某些 Web service 而不用对使用这些 Web service 的客户端的知识有任何了解。

目前，Web service 是最适合实现 SOA 的一些技术的集合，它为广泛实现 SOA 架构提供了基础。SOA 的主要目标包括以下一些：

① 重用与合成：能够在应用之间重用模块并能在内部应用之间实现互交换；

② 松散耦合：把服务使用者和服务提供者在服务实现和客户如何使用服务方面隔离开来；

③ 灵活性：每一个应用都有一定的生命周期，因此它应该可以被重新纳入新的模块，支持新的业务需求；

④ 开放性和互操作性：以便能在不同平台和环境下共享模块；

⑤ 分布式：使模块可以被远程访问并被集中管理。

HLA 和 SOA 是实现互操作的两种典型体系结构，但两者产生的背景不同。SOA 产生于企业的 IT 系统和商业流程，而 HLA 源于军事应用，因而两者实现互操作的机理不尽相同，SOA 主要是基于功能的重用，SOA 的关键是“服务”的概念。HLA 主要是基于模型的重用，它采用智能总线(RTI)技术、“发布”和“订购”机制以及采用统一的数据交换格式

(XML)来实现大规模系统的集成。

基于 HLA 进行仿真虽然也可以实现一定程度上的重用,但是对资源的管理却缺乏统一的标准,而 SOA 提供了一种相对完善的资源重用、共享的架构,基于 SOA 进行仿真可以更方便地实现仿真资源的管理、重用。两者在系统集成上各有千秋,将两者结合起来,取长补短,就成为一个很好的选择。

目前,两者结合有两种模式:迁移模式和统一架构模式。

迁移模式是通过一个所谓的连接器的模块来实现将现有的基于 SOA 的系统与基于 HLA 的系统进行集成,如图 11.21 所示。

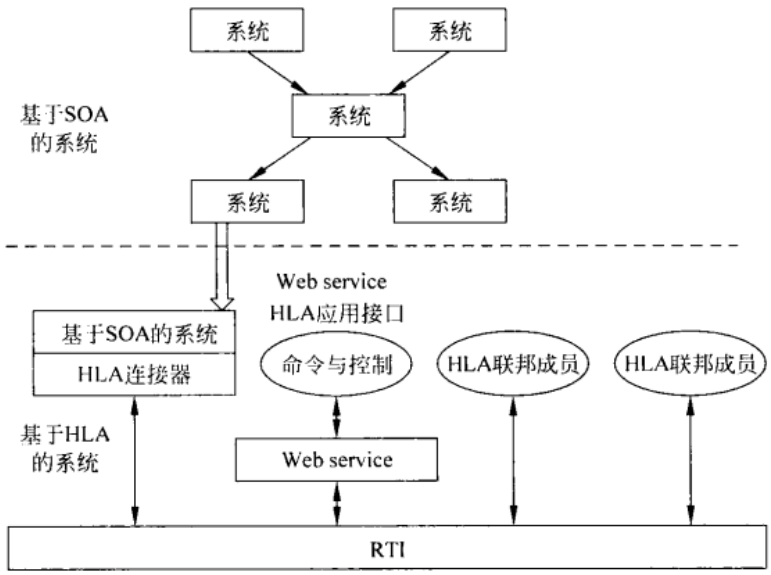


图 11.21 HLA 连接器

另一种是统一架构模式,在兼收并蓄 HLA 和 SOA 的基础上,构建一个全新的架构模型,如图 11.22 所示。

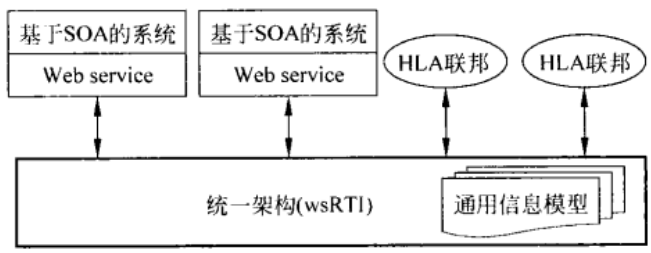


图 11.22 统一架构模型

11.5.3 仿真网格(SimGrid)

1. HLA 的不足

高层体系结构(HLA)是现在分布式仿真的主要标准,HLA 在互操作性和重用性方面

有了很大提高。但是，随着仿真规模的扩大和要求的不断提高，一些问题已经制约了 HLA 的发展，具体表现在：

1）资源的动态分配较为困难

在基于 HLA 的分布式仿真中，仿真任务一般都不是动态分配的。仿真一旦开始，各仿真节点都执行各自的仿真任务，网络的主要作用是在各仿真节点之间传输交互数据。各节点的仿真任务都与节点捆绑在一起。正是这种捆绑造成了任务不能根据各节点的实际情况进行动态分配，使得某个仿真节点一旦出现故障，将导致整个仿真任务不能顺利进行。由于每个仿真节点执行的任务各不相同，当有仿真节点出现故障（网络不通或者仿真计算出现软硬件故障）时，该仿真节点的仿真任务就不能完成；如果主要的仿真节点出现故障，就会造成整个仿真的瘫痪。资源的静态分配不利于合理有效地利用资源，而 HLA 标准中并不提供分布交互仿真运行时所需资源进行动态管理的机制。

2）资源动态的负载平衡较为困难

仿真运行时的过程由仿真设计静态地固定下来，对仿真系统突发的事件缺乏动态的适应能力，因此资源动态的负载平衡较为困难。传统仿真系统中，资源分配是在仿真设计时规定的。仿真过程一旦规定下来，在仿真运行阶段将不会改变。例如，由于某种原因某个节点的负载突然增大，整个系统往往不会随之调整运行方式（如把该节点的运行任务迁移到别的节点上去），从而造成系统效率大大下降。

3）仿真资源缺乏统一、标准的访问接口

分布式仿真所涉及的资源种类繁多，类型各异。资源不同，访问的接口也不同，缺乏一个统一的、标准的访问接口。仿真应用在使用这些资源的时候必须要针对不同的接口设计不同的访问过程，为仿真应用开发带来了许多额外的工作和开销，同时这也不利于仿真资源的重用。

4）缺乏对大型的并行仿真应用的支持

现有的仿真系统中的仿真应用通常采用串行的程序处理方式。但某些仿真应用的计算量非常巨大，采用串行的程序处理方式往往不能够适应。这时，通常采用简化仿真模型的方式来减少工作量。但这样做将相应地降低仿真的精度和可信度。如果使用并行的程序处理方式，将不可避免地使用大型的计算机来提供计算资源。通常，大型计算机的计算资源将给多个仿真用户共享使用。

2. 网格的基本概念

网格是继万维网之后出现的一种新型网络计算平台，网格一词最早出现在 20 世纪 90 年代中期，是从电力网概念借鉴过来的。网格的最终目的，是希望大家能够像使用电力一样方便地使用分布在网络上强大而丰富的计算能力。网格技术是一项正在快速发展的新兴技术，其核心是解决网上各种资源（如计算资源、存储资源、模型资源、数据资源等）的动态共享与协同应用。

网格有以下四个方面的特点：

1）分布与共享

分布性是网格的一个最主要的特点。网格的分布性是指网格的资源是分布的，涉及的资源类型复杂、规模较大、跨越的地理范围较广。在网格这一分布式环境下，需要解决资源

和任务的分配与调度问题，安全传输与通信问题，实时性保障问题，人与系统以及人与人之间的交互问题等。

网格资源虽然是分布的，但是它们却是可以充分共享的，即网格上的任何资源都可以提供给网格上的任何使用者。共享是网格的目的，没有共享便没有网格，解决分布资源的共享问题是网格的核心。分布是网格硬件在物理上的特征，而共享是在网格软件支持下实现的逻辑上的特征，这两者对于网格来说都是十分重要的。

2）自相似性

自相似性在许多自然和社会现象中大量存在，一些复杂系统具有这种特征，网格也是这样。网格的局部和整体之间存在着一定的相似性，局部往往在许多地方具有全局的某些特征，而全局的特征在局部也有一定的体现。

3）动态性和多样性

网格的动态性包括动态增加和动态减少两方面含义。对于网格资源的动态减少或者资源出现故障的情况，要求网格能够及时采取措施，实现任务的自动迁移，做到对高层用户透明或者尽可能减少用户的损失。资源的动态增加要求网格具备可扩展的能力，允许新的资源加入到网格中，与原来的资源融合在一起。网格的扩展性要求体现在规模、能力、兼容性等几个方面。

网格资源是异构和多样的，在网格环境中可以有不同体系结构的计算机系统和类别不同的资源，网格系统必须能够解决这些不同结构、不同类别资源之间的通信和互操作问题。

4）自治性和管理的多重性

网格资源的拥有者对该资源具有最高级别的管理权限，网格应该允许资源拥有者对他的资源有自主的管理能力，即网格的自治性。此外，网格资源也必须接受网格的统一管理，建立不同资源相互之间的联系，实现共享和互操作。

3. 仿真网格

仿真网格是在新兴网络技术（如互联网技术，Web service 技术和网格技术）与传统网络化建模与仿真支撑技术（如 HLA 技术）的结合下，诞生的新一代网络化建模与仿真支撑系统，旨在克服现有仿真技术的不足，使各种满足 HLA 标准的分布仿真系统在计算网格环境下，充分利用网格技术优势，满足越来越复杂的仿真需求，并最终提高分布仿真系统的效率和容错性，减少系统开发、维护和使用的时间及成本，提高系统的验证性、有效性和置信度。仿真网格具有如下的技术特征：

（1）实现仿真网格/联邦中各类资源安全地共享与重用，动态调度运行与优化，支持复杂仿真应用系统基于网格环境的快速构造和开发。

（2）实现仿真网格/联邦中各类仿真模型/模拟器/仿真仪器/实物设备/人之间跨组织、跨领域、异构、分布、安全的互操作和协同仿真。

（3）支持虚拟团队/组织、过程、仿真模型资源和仿真应用工程项目基于网格的管理与优化。

（4）提供面向应用、用户友好的各类问题求解环境/网格门户和基于网格的 VR/可视化显示环境；提供面向仿真应用领域的、支持仿真应用系统全生命周期协同开发应用的各类支撑工具和建模仿真工具集。

(5) 支持以异地、协同虚拟设计/虚拟制造和大规模体系级仿真为代表的紧耦合/协作型(资源/服务组成一个作业,资源/服务间有信息流、事件流,强调时空一致性与交互操作等)协同工作;也能支持以异地、协同设计/生产/管理为代表的松耦合型协同工作;多类(高性能)资源/服务共享与协同;多用户同时工作。

(6) 扩展了分布仿真应用的模式,实现了基于 Internet 和网格的协同仿真应用新方法,拓展了网格的应用领域,弥补了传统计算网格、数据网格在协同工作支持上的不足。

4. 基于 HLA 的仿真网格构建技术

基于 HLA 构建的仿真系统在国内外都有很多成熟的应用实例,将网格技术与 HLA 结合构建仿真网格,国内外都有不少研究,下面介绍两种典型方式。

1) 基于服务的 HLA/RTI 的仿真网格

如图 11.23 所示,将 RTI 的六大服务(联邦管理服务、声明管理服务、对象管理服务、时间管理服务、所有权管理服务和数据分发管理服务)封装成服务。

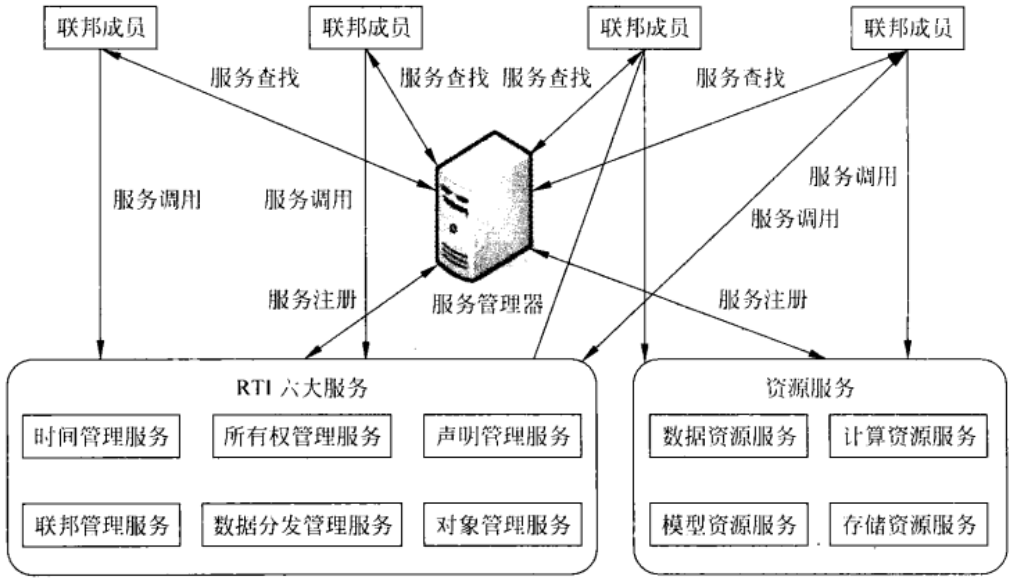


图 11.23 基于服务的 HLA/RTI 的分布式仿真结构图

这种方式下把 RTI 作为一种资源,可以在整个网格环境中实现部署和运行。仿真开发人员可以在仿真网格中动态调用网格环境中的各种仿真资源服务构建仿真系统。但是,这种方式下 RTI 联邦成员调用 RTI 服务时响应时间比较长,联邦成员之间数据交换通过服务的方式,系统延迟比较严重,系统实时性比较难保证,而且原有基于 HLA 构建的仿真系统移植到基于服务化 RTI 的仿真网格工作量比较大。

2) 基于传统 HLA/RTI 的仿真网格

如图 11.24 所示,这种方式下,各种仿真资源被封装为服务后,动态部署在仿真网格节点的服务容器中,并注册到服务管理器。仿真开发人员可以利用服务管理器提供的查找功能在仿真网格中查找和定位各种仿真服务,构建仿真系统。仿真系统运行时动态调用网格环境中的各种仿真服务,完成仿真系统的运行。这种方式下,HLA 负责联邦执行的创建、

联邦成员的加入、联邦成员之间数据的交互等，没有改变现有 HLA/RTI 标准，原有 HLA 仿真系统很容易移植到仿真网格中。联邦成员之间的数据交互通过 HLA/RTI 提供的公布/订阅机制，数据传输量比较小，减少占用网络带宽。

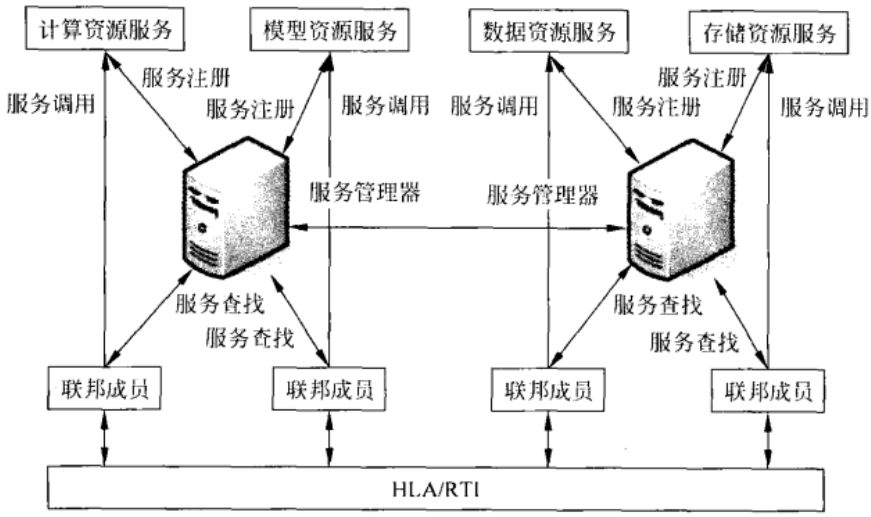


图 11.24 基于传统的 HLA/RTI 的仿真网格结构图

11.5.4 小结

如前所述，目前国外进行分布式仿真研究的组织和机构较多，研究情况可以分为以下三类：

(1) 基于 Web 服务的构建仿真平台技术研究，其研究内容是将 Web 技术应用到仿真领域中，在 Internet 上进行仿真资源共享的研究，构建广域的资源共享平台，典型代表是美国国防部的 XMSF 仿真建模框架。Morse 等人在 XMSF 的研究中提出了一种基于 Web service 的 HLA 体系结构，该体系结构实现了基于 Web 的联邦成员的 RTI 通信，通过 SOAP 对 RTI 调用标准化。

(2) 基于并行计算的仿真技术研究，以 SPEEDS 项目为典型代表。SPEEDS 是一个非常复杂的系统，主要用于计算密集型的仿真。目前它已经用于多个大型的军用仿真项目中，如 DoD 的 JSIMS 等。

(3) 基于网格技术构建仿真平台技术研究，其中典型的代表是欧洲的 CrossGrid 研究项目。该项目中明确提出了将网格技术与仿真结合的研究方向，以及使用网格服务构建基于 HLA 仿真应用的方案，并提出在基于(open grid service architecture，OGSA)的网格平台中，基于网格构建 HLA 仿真应用的三个层次的解决方案：将 HLA 的仿真应用通过 RTI 层次的移植、联邦成员层次的移植、联邦层次的移植三个层次移植构建基于网格的 HLA 仿真应用。新加坡的南洋大学也进行基于网格的分布式仿真的研究工作，在 2002 年，他们提出了在目前 HLA 体系中添加一个(load management system，LMS)的结构，LMS 负责为基于 HLA 的仿真任务分配资源和联邦运行出现故障时进行一些应急的处理，保障仿真任务的顺利进行，通过 OGSA 的网格服务解决了为异构仿真联邦成员提供 HLA/RTI 支撑

问题。

此外，还有一些基于网格的仿真项目，例如美国地震工程仿真网格 NEESGrid 和用于复杂系统仿真研究与应用的 SF express 项目等。

总之，关于仿真网格的研究得到广泛重视，目前还在进一步发展中。

参考文献

[1] 陈国良. 并行算法的设计与分析. 修订版. 北京：高等教育出版社，2002

[2] Baker K D. Development of Cellular Multi-microprocessor Systems. IEEE Conference Proceedings, 1977,3(6)

[3] 谢廿弟. 多机并行处理系统国内外研究发展情况及动向. 系统仿真，1987,(2)

[4] 张晨曦. 连续系统仿真专用多处理机的探讨. 系统仿真，1987,(2)

[5] 姚小民，龙卫红. 并行仿真计算的任务划分问题研究. 系统仿真学报，1994，6(2)：47～51

[6] Jefferson D, et al. Distributed Simulation and the Time Warp Operating System. Proc. Symposium. Operating Systems Principles, Austin, Texas,1987

[7] Madisetti V, Hardaker D and Fujimoto R. The Mimdix Operating System for Parallel Simulation [C]. Proc. Sixth Workshop on Parallel and Distributed Simulation, SCS Western Simulation Multi-Conference on Parallel and Distributed Simulation, SOC. For Computer Simulation Int'l, San Diego, Calif. ,1992：65～74

[8] 费景高. 微分代数方程的一类并行算法. 计算机科学与工程，1989,(5)

[9] 斯普里特 J A 等著. 王正中等译. 计算机辅助建模与仿真. 北京：科学出版社，1991

[10] Katz W. Distributed Interactive Simulation-Interoperability is Key to Success in Simulator Networking. Defense Electronics,1994,4：10～13

[11] 周绥平，陈宗基. 分布交互仿真环境研究. 系统仿真学报，1995,7

[12] 王行仁等. 分布交互仿真. 系统仿真学报，1995,7

[13] 周绥平，陈宗基. 分布交互仿真的时空一致性. 系统仿真学报，1996,8(3)

[14] 刘秀罗，薄涛. 新一代分布式交互仿真支撑软件——RTI. 计算机应用，2001,21(4)

[15] 郭齐胜，张伟，杨立功等. 分布交互仿真及其军事应用. 北京：国防工业出版社，2007

[16] 付正军，王永红译. 计算机仿真中的 HLA 技术. 北京：国防工业出版社，2003

[17] Bjorn M., Staffan L.. Mixing Service Oriented and High Level Architectures in Support of the (Global Information Grid)GIG. 2005

[18] 张兵. 基于 OGSA 与 HLA 的仿真网格环境构建技术研究[硕士学位论文]. 南京：南京理工大学，2007

[19] Morse K L, Drake D L, Brunton P. Web enabling an RTI—an XMSF Profile. In：IEEE 2003 EUROPE AN SIMULATION INTEROPERABILITYWORKSHOP,2003

[20] Furmanski W. SPEEDES Installation and Training at ERDC MSRC. ERDC MSRC PET Technical Report, 2001, 5

[21] Kunze M. The CrossGrid Project. In：Acat 2002, Moscow, Russian Federation, Apr 21 2003,382～385

[22] Gomes J, et al. Experience with the International Testbed in the CrossGrid Project. In：European Grid Conference on Advances in Grid Computing-EGC2005, Amsterdam, Netherlands,2005,98～110

第 12 章
虚拟现实技术*

我们知道，传统的冯·诺依曼计算机要想充分发挥其功能，必须遵循“可计算性”的三个前提与条件，即：系统模型描述的形式化；一个可执行的算法；合理复杂度的算法实现。计算机可接收的是数字化信息。为了提高计算机的效能，人们进行了大量的努力，其中最主要的是所谓两个“M”和两个“O”。两个“M”是指“大规模并行处理技术”(massively parallel processing)和“多媒体技术”(multimedia)，而两个“O”是指“面向对象技术”(object-oriented)和“开放系统技术”(open system)。然而，这并未从根本上克服冯·诺依曼计算机所要求的“可计算性”的前提和条件，以满足人们所希望的“以人的感知和认知方式表示和处理信息”的要求。未来的计算机系统应不仅仅是建立在单一的数字化信息的基础上，而应基于多维化的信息空间。在这种需求的驱动下，虚拟现实技术从 20 世纪 80 年代开始蓬勃地发展起来。

12.1 虚拟现实的特征及基本构成

什么是虚拟现实？它具有哪些主要特征？其基本构成是什么？本节进行概括性说明。

应该说，尽管不少文献对虚拟现实进行过定义，但到目前为止，这些定义仍缺乏公认的权威性，这恐怕要归咎于虚拟现实技术发展的历史还不够长，许多技术还不够成熟。尽管如此，为了便于说明人们目前对虚拟现实的认识，参考国内外的典型文献，我们仍给出如下定义：

虚拟现实(virtual reality，VR)是一种基于可计算信息的沉浸感交互环境，在该环境中，人们可以创建和体验虚拟世界(virtual world)。虚拟世界是全体虚拟环境(virtual environment，VE)或给定仿真对象的全体。虚拟环境是由计算机和电子技术生成的，通过视、听、触觉等作用于用户，使之产生身临其境感觉的交互式视景仿真。

因此，从根本意义上来看，可以将虚拟现实技术视为交互式仿真技术的高级形式。与传

统的一般交互式仿真的主要区别在于：

① 信息多维性。VR 基于多维信息而不仅仅基于数字信息，包括声音、图像、图形、位姿、力反馈、触觉等。

② 人机交互的自然性。传统人机交互系统借助于键盘、鼠标或专用的控制设备（如各类训练仿真系统），用户发出指令，机器执行指令，强调人适应计算机。VR 强调的是计算机适应人，计算机可以识别人的位姿、手势，甚至人机可以“会话”，头盔、数据手套、数据衣等成为人机自然交互的基本手段。

③ 理想的 VR 应达到人在虚拟环境中如同在真实环境中“一样”或“接近一样”的感觉，即除了三维视景感觉外，还具有自动定位的听觉、触觉、力觉、运动等感知，甚至具有味觉、嗅觉等。

④ 计算机系统除了计算功能外，还能表现出具有知识、智能等功能。

总的来说，实物虚化、虚物实化和高性能的计算处理技术是 VR 技术的 3 个主要方面。

实物虚化是现实世界空间向多维信息化空间的一种映射，主要包括基本模型构建、空间跟踪、声音定位、视觉跟踪和视点感应等关键技术，这些技术使得具有真实感的虚拟世界的生成、虚拟环境对用户操作的检测和反应，以及操作数据的获取成为可能。

虚物实化是指确保用户从虚拟环境中获取同真实环境中一样或相似的视觉、听觉、力觉和触觉等感官认知的关键技术。能否让参与者产生沉浸感的关键因素除了视觉和听觉感知外，还有用户能否在操纵虚拟物体的同时，感受到虚拟物体的反作用，从而产生触觉和力觉感知。

高性能计算处理技术主要包括数据处理，实时、逼真的图形图像生成与显示，多种声音的合成与声音空间化，多维信息数据融合、数据压缩以及数据库的生成等，以实现命令识别、语音识别以及手势和人的面部表情信息的检测等在内的模式识别，大规模的远程网络分布式与并行计算等。

Burde G. 在 Electro'93 国际会议上所发表的“虚拟现实系统及应用”（virtual reality systems and application）一文中，曾提出“虚拟现实技术三角形”，简捷地说明了虚拟现实系统的基本特征，即三个“I”，它们是：immersion-interaction-imagination（沉浸-交互-创意）（参见图 12.1）。

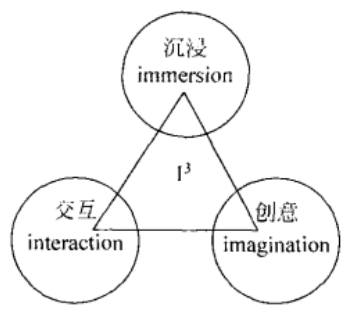


图 12.1 VR 的基本特征

沉浸——VR 追求的目标是力图使用户在计算机产生的三维虚拟环境中有身临其境感，在环境中的“一切”，看上去是真的，听起来是真的，动起来是真的，用户觉得自己是环境中的一部分，如同在已有经验的现实世界中一样，而不是旁观者。

交互——VR 与用户的交互是三维的，用户是交互的主体，且是多感知的。

创意——过去用户只能在以定量计算为主的结果中加深对对象的认识，VR 则可使人从定量和定性两者的综合中得到感性和理性的认识，从而得到启发、深化概念和萌发新意。

根据上述 VR 的定义及其基本特征，可以看到，VR 系统是一个十分复杂的系统，它所涉及的技术包括计算机图形学、图像处理与模式识别、智能接口、人工智能、多传感器、语音处理与音响、网络、并行处理、系统建模仿真、系统集成等。为了实现这些基本特征，VR 的

结构可用图 12.2 表示。

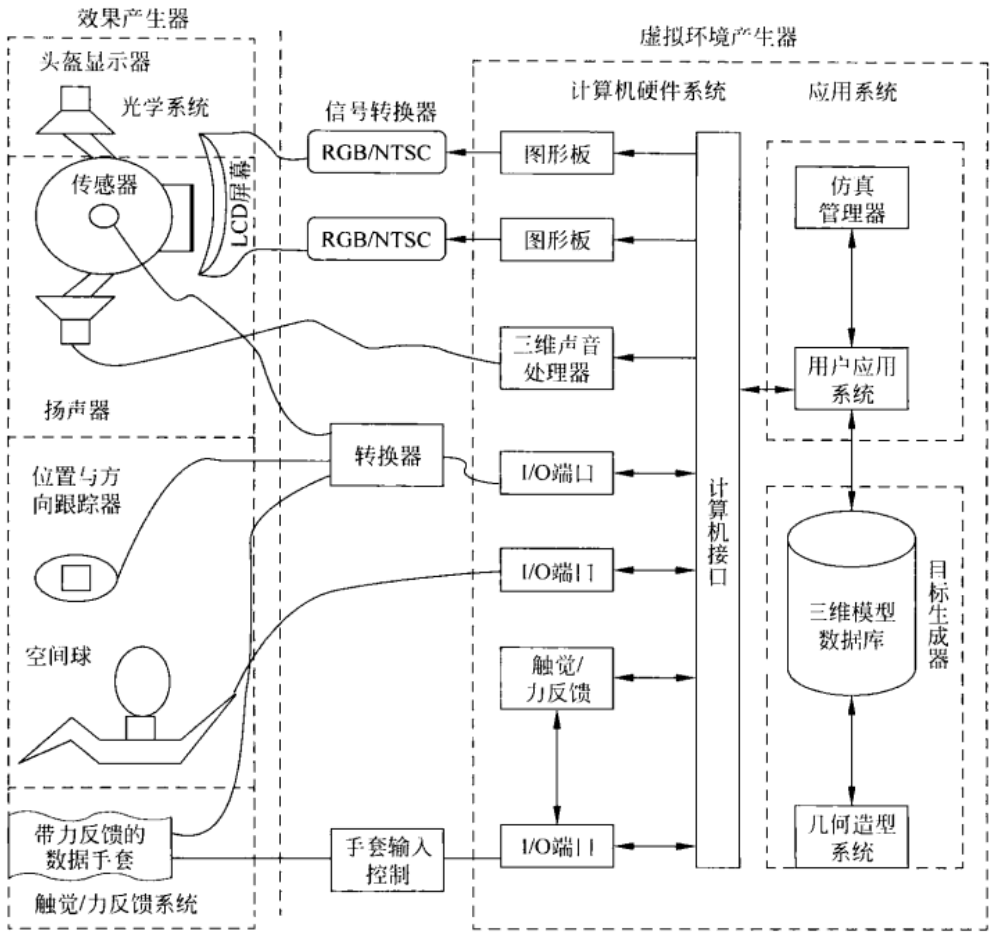


图 12.2 典型 VR 系统的组成

从图中可以看出，典型的虚拟现实系统由以下几部分构成。

(1) 虚拟环境产生器

这是虚拟现实系统的主要部分，其目标是为用户产生虚拟环境并实现运行管理。它由两部分构成，即应用系统和计算机系统。

应用系统包括两部分。一部分由几何造型系统和三维模型数据库构成，也称为目标生成器，这是通过三维几何造型系统按照应用要求离线生成的，对仿真对象的物理属性(外形、颜色、位置等)进行建模，为图像着色、添加纹理、阴影，对图像进行剪裁、消隐、添加光照等。另一部分由用户应用系统和仿真管理器组成，用户应用系统是用于定义对虚拟现实系统进行操作的内容，如仿真的动态逻辑、结构，以及仿真对象间和仿真对象与用户间的交互关系。仿真管理器负责用户使用虚拟现实系统时的管理，它基于计算机操作系统，实现仿真过程的任务、进程、资源、对象等所对应的场景、事件、运动等协调，使使用者如同进入真实环境中。

应用系统大多是完成上述功能要求的软件系统，它当然离不开计算机硬件系统。用于 VR 系统的计算机系统一般都具有高速的运算能力、图形处理能力和输入输出能力，甚至还

需要并行处理能力。因此，大容量的内存、外存，图形加速板，高速的I/O接口板是VR系统中计算机配置的基本特征。另外，为了能接收各类专用设备的输入信号，往往还包含各种特殊的接口板。

(2) 效果产生器

效果产生器主要包括头盔显示器、位置与方向跟踪器、三维声音处理器及触觉/力反馈装置等。

头盔显示器具有多方面的功能，其目的主要是使用户增加“沉浸”感。一般固定在人的头部，用两个LCD或CRT显示，分别像两只眼睛显示由虚拟环境产生器产生的图像。这两个显示器中的图像是由计算机的两个图形板分别驱动的，屏上的两幅图像存在细小的差别，类似于我们的“双眼视差”，大脑将融合这两幅图像获得“深度”视觉。

为了与虚拟环境交互，必须感知参与者的位置，特别是要感知参与者的视线，即跟踪头部的位置及方向，位置与方向跟踪器在VR系统中一般是不可缺少的，而且，跟踪器将感知的信息送入计算机，应能快速地更新图像(一般延迟时间不能大于0.1s)，否则用户将产生很不舒服的感觉。在方向跟踪方面，眼球位置变化(特别是方向变化)的跟踪是最难以解决的，目前尚无有效的设备。在虚拟环境中飞行和漫游时往往使用空间跟踪球，大多数桌面VR系统采用这种方式定位。“数据手套”是虚拟现实技术中的典型设备。它是一种可戴在用户手上的传感装置，可监视手的位置和方向、手指的弯曲(包括手指关节的角度变化、各手指之间及手腕之间的相对运动)等，因而也属于位置跟踪设备。

三维声音处理器包括声音合成、三维声音定域和语音识别。虚拟现实的临场感通常需要立体声，为此，需要设置静态和动态声源，并构成动态声学环境。在VR的应用中，声音定域十分重要。声音定域系统要能实时捕捉声音的发生源和头部的相对位置和方向，从而实时地分别在左右耳产生出恰当强度的立体声，这对提高“沉浸”感和“创意”是十分重要的。例如，当看到一辆火车奔驰而来的时候，不仅图像要实时变化，声音也应自小到大变化，否则就不会有“真实感”。语音识别的目的是让计算机具有人类的听觉功能，这也是目前VR需要着重解决的难题之一。

属于触觉/力反馈装置最常见到的是计算机键盘、鼠标、空间球以及对游戏者的施力作出反应的游戏杆，这是相对简单的触觉/力反馈通道，数据手套和数据外衣是比较复杂的触觉/力反馈通道。特别是某些数据手套，它可以抓取虚拟物体，并测量出虚拟物体的反作用力，从而实现力反馈。触觉识别装置进展缓慢，这也是目前VR需要着重解决的另一难题。

(3) 接口设备

接口设备既包括硬设备，也包括软设备(接口软件)，也有文献将它们分别归类到效果产生器或虚拟环境产生器。如将计算机产生的图像转换成LCD图像的转换器，位置与方向跟踪系统的信号转换器，以及数据手套的输入控制设备等。

总之，VR涉及的内容非常广泛。由于篇幅的限制，本书只能就VR的一些主要技术进行初步介绍。在以后的几节中，我们将分别讨论以下内容：首先就VR技术的发展历程进行概括性介绍，然后讨论VR中的基本技术，如位置跟踪通道、视觉通道、听觉通道、触觉/力反馈、虚拟场景的生成等，最后对虚拟现实技术进行简短的小结。

12.2 虚拟现实技术的发展概况

12.2.1 虚拟现实技术的发展历程

虚拟现实技术的历史可追溯到20世纪50年代开始出现的立体电影以及各种宽银幕、环幕、球幕电影。具有深度感、大视野、环境感的电影图像加上音响配合，使观众“沉浸”在电影之中。20世纪60年代初，电影摄影师 Morton Heiling 构造了第一个多感知仿真环境 sensorama simulator。这实际上是第一个 VR 视频系统，观众坐在拱顶装置下，手操摩托车把，“穿行”于纽约闹市，观察到各种街景，并伴随各种车水马龙的立体声，感觉到车行的颠簸等。

在工程界，虚拟现实技术起源于飞行模拟器。尽管第一个飞行模拟器（利用 Edwin Link 在自动钢琴演奏机上的气压随动装置原理设计）的视觉很不直观，但随着计算机的功能和计算机图形技术的发展，飞行模拟器的视景系统从模拟式闭路投影电视发展到数字成像，训练人员在坐舱可以观察到起飞或降落时机场跑道的视景变化，但不能随观察位置变化而改变视景。

1965年，有 VR“先锋”之称的计算机图形学的创始人，当时任 ARPA（美国国防部高级研究计划署）信息处理技术办公室主任的 Ivan Sutherland 发表了题为“终极的显示（The ultimate display）”的论文，提出了如何把计算机显示屏作为“一个通过它观看虚拟世界的窗口”，使观察者有身临其境感觉的思想。这被认为是虚拟现实概念的开端。1968年，他领导研制了第一台头盔式显示器，尽管很笨重，不得不采用机械手臂来支撑，但成为三维立体显示的奠基性成果。

1971年，Frederick Brooks 研制了具有力反馈的装置 grope Ⅱ，用户可通过操纵一个实际的机械手，去控制屏幕中的虚拟机械手“抓取”屏幕中的虚拟物体，用户的手有重量感觉。1975年，Myron Krueger 提出了“人工现实（artificial reality）”思想，他构想了一种头盔显示器 HMD（head mounted display），由三部分构成：能像眼镜一样放在眼睛前的液晶显示器，能像耳机一样放在耳旁的立体声输出装置，以及能跟踪观察者头部位置的跟踪装置。他还设计了双目潜望镜式的全景器（binocular omni-oriented monitor，BOOM），观察者可透过观察孔观察到类似全景式的图像。HMD 与 BOOM 的显示图像来自计算机输出或遥控闭路摄像机的输出。

总体说来，20世纪60年代到80年代中，VR 技术的发展是十分缓慢的，这主要受限于计算机技术。1985年，美国航空航天局（NASA）的 Scott Fisher 研制成功“数据手套（dataglove）”，这种具有柔性、轻质的手套装置可以量测手指关节动作、手掌弯曲等；1986年，被认为是第一个可以称得上“完整 VR 系统”的（virtual interactive environment workstation，VIEW）研制成功，使用了头盔、显示器、数据手套、语言识别、跟踪技术等，并应用于空间技术、科学数据可视化、远程操作等领域。因此，NASA 被视为 VR 的发源地。自此以来，VR 技术得到迅速发展。

1989年，VPL 公司的创始人之一 Jaron Lanier 首先提出“虚拟现实（virtual reality）”这

一概念，引起新闻媒介极大关注，于是，VR 便不再是研究部门科技人员的专用名词，成为社会广泛关注的课题。

那么，VR 技术与传统的仿真技术有何联系又有何区别呢？首先，两者都是对现实世界的模拟，都力图通过计算机及各类装置达到现实世界尽可能精确地再现；两者都是基于模型的活动。虚拟现实技术是在仿真技术基础上发展起来的，从这个角度来说，VR 是仿真的高级阶段。按照美国国家科学出版社(National Academy Press)1995 年出版的“虚拟现实(virtual reality)”一书的解释，虚拟现实的虚拟环境(virtual environment，VE)系统与仿真器系统的区别是很细微的，仿真器系统或多或少正在向 VE 系统演变，一般说来，在以下情况下，我们称系统为 VE 系统，而不称其为仿真系统。VE 系统具有以下特征：

① 系统可在软件上方便地重新配置；

② 系统可用于创造各种非自然环境和自然环境；

③ 系统具有良好的交互性和适应性；

④ 系统应用了许多人类感知模式和感受动作系统；

⑤ 用户高度融入计算机合成环境，在人工环境中有身临其境之感。

12.2.2 虚拟现实技术发展的瓶颈

尽管人们预言 VR 将在不少领域得到应用，但目前最多的是在娱乐方面。大多数系统只有视觉和听觉的感受，极少涉及嗅觉和味觉，触觉面临极大的困难，力反馈装置尽管有一些研究成果，但离实际应用还有较大的距离。需要解决的主要技术问题包括：

① 实时图形与图像技术

虚拟现实要求图形与图像的刷新速度至少不低于 15 帧/s，图形客体行为反映的滞后时间要小于 0.1s。研究表明，人眼所能分辨的图像可用 8000 万个多边形来显示，而现代的计算机，像 SGI 的 onyx reality engine 只能每秒处理 200 万个平面三角形或 90 万个带纹理的三角形。即使像目前最先进的北卡罗来纳大学图形处理器 pixel flow 每个每秒只能处理 140 万个带纹理的三角形，只有当 128 个 pixel flow 并行处理才能达到 1 亿个三角形。何况还需要进行其他操作运算。

② 位置跟踪与映射技术

手的跟踪：据研究，一般手腕的切线速度可达 3m/s，而加速度为(5～6)g，手臂运动时的运动带宽约为 2Hz；如果手的运动是用来驱动遥控机器人或进行手臂的动态仿真，考虑到如感应器噪声等因素，那么采样频率要达到带宽的 20 倍左右，大约为 40Hz。

眼睛的跟踪：眼睛的运动最快为 600°/s。眼睛扫视的最短时间常数约为 50ms；最小的扫视可以在 60ms 内完成。采样频率必须是测量噪声带宽的 20 倍，眼睛运动推荐的采样频率是 1kHz，当眼睛运动的采样频率为 1kHz 时，被显示物体能够很好地以每六十分之一秒显示一幅。

头部的跟踪：头部运动最快可达 1 000°/s，而通常最快的速度是头部晃动为 600°/s 和点头摇摆为 300°/s，因此跟踪器向主机报告的速率不能低于 30Hz。我们已经知道头部运动和虚拟反馈之间的延迟大于 60ms 将会损害适应性和幻像的显示，对于好的系统不应该超过 10ms。为了达到完美的、平滑的、稳定的运动，分辨率必须达到对方向为 0.03°，对传递为

0.01mm。为达到上述要求，各类跟踪设备还需要在性能方面做重大改进。

③ 触觉/力反馈

人体通过触觉感可以辨别出 0～1kHz 的振动。手指的单点空间分辨率为 0.15mm，双点空间分辨率的阈值为 1mm。人对光滑平板的检测阈值为 2μm 高的单点和 0.06μm 高的栅格。

从根本上来讲，触觉模型的研究尚处于起步阶段，如何将手指的位置跟踪与触摸模型结合尚待研究。

④ 人的心理因素

VR 系统的特点是真实的人与虚拟的环境交互，人的心理因素对 VR 的效能评价起着至关重要的作用。人的感知过程、认知过程、大脑神经的作用等，对虚拟环境往往都用“像”或“不像”来描述，甚至哪些技术上的改进能提高“逼真度”都难以确定。加之，不同的人对虚拟环境的适应程度、接受程度差异很大，这更增加了 VR 系统实现的难度。

⑤ 系统集成

到目前为止，虚拟现实技术还缺乏统一的技术标准，而虚拟现实系统所需集成的设备、接口、软件等种类繁多，这给系统集成带来巨大困难。一个最基本的问题，采用 RS-232 接口通信已远远不能满足 VR 中外部设备与计算机之间交换数据速度的需要，到底采用何种接口标准为宜仍在研究中。

总之，虚拟现实技术涉及的技术领域十分广泛，现有的技术手段及可见的技术手段远远不能满足需要。虽然 VR 技术具有广阔的应用前景，但目前尚处于“婴儿”阶段，还存在着很多技术难题和理论问题亟待解决。

12.2.3 当前流行的虚拟现实系统

1. 桌面系统

这类系统沿袭了伊凡·萨瑟兰最初提出的“把计算机显示器作为观察虚拟世界的窗口”这一思想，所以，又被称为“世界之窗”。其特点是只用普通个人计算机的显示器显示虚拟环境，以鼠标、键盘作为交互设备来控制虚拟环境中的运动。用户能得到有一定层次效果的三维图像，只是系统的三维空间感稍差。但因其价格适中，相关技术在个人计算机上即可实现，是当前在娱乐、工程、建筑领域应用最广的虚拟现实系统。随着计算机硬件设备性能的飞速提高，它们大有赶超沉浸式系统，得到更广泛的普及的势头。

2. 沉浸式系统

这类系统的特点是用封闭的视景和音响系统将用户与外界隔离，使之完全沉浸在计算机产生的三维世界中。它需配备特殊的交互设备，如头盔显示器、数据手套、跟踪和触觉反馈系统，来共同营造身临其境的高度沉浸感。这是虚拟现实传统的研究方向，但是高昂的造价、沉重的器械负担和技术的复杂性，使它大部分只存在于一些科研机构和大企业的实验室里。

3. 投影式系统

在计算机系统的支持下，通过头盔式显示器，使人们能够获得身临其境的沉浸感。但是头盔也带来了一个缺点，那就是当许多人都佩戴上头盔显示器后，他们每个人都基本上与外界的真实环境隔离了。如果他们想与周围的合作伙伴讨论问题和交流就很不方便，而且现在的头盔显示器也不完善。因此，科技人员又致力于开发一种能多人同时观看的、具有高分辨率又能获得沉浸感效果的 CAVE 系统。

CAVE 是由美国伊利诺伊大学的电子可视化实验室在 1992 年研制成功的。它是一个 10ft×10ft×10ft(1ft=0.304 8m)的特制房间，四周是背投式的投影墙，下方是正投式的投影地面。研究人员戴上立体眼镜进入 CAVE 后，就完全沉浸在由前后左右 4 个方向以及地面(下方)的图像所营造的虚拟环境的氛围中了，可以身临其境地观看、听闻、触摸这里的虚拟对象和虚拟环境。在这个特殊装置的周围，配备了高性能的计算机系统，并且提供了与所研究问题有关的大量的数据、图形、模型、工具和辅助决策系统。当研究人员进入 CAVE 时，可以用一个称为“魔棍”的手持三维鼠标跟虚拟空间发生交互作用，用它来控制、引导虚拟对象的运动和虚拟景物的切换。这类系统的特点是佩戴设备轻便、自由度大、沉浸度高，适宜作多用户的教学演示系统。这是较理想的虚拟现实系统，但因技术和成本的问题，还没有成为研究的主流。近年来又研制出小型化的 ImmersaDesk 系统，它只有一个投影面，但分辨率高，占地小，得到了一定程度的推广。

4. 分布式系统

这是虚拟现实技术和计算机网络通信技术的结合、合理地利用计算机资源，它不但能充分建立起“面对面”，而且用网络连接在一起的不同用户的工作环境，实时共享同一虚拟空间，以达到相互感知、协同工作的目的。它代表了发展的崭新方向，但兼容性、受限带宽、延迟仍是当前主要的技术难点。

5. 增强型虚拟现实系统

增强型虚拟现实是近年来新发展起来的一门技术，它是一种虚拟的世界与真实的环境结合在一起的虚拟现实系统。

它利用计算机产生的图形或文字信息，对周围真实世界的场景动态地进行增强，使用户在看到真实环境的同时，也同步看到计算机产生的增强信息。这样，当用戴上数据手套的手去搬动一个虚拟的操纵杆时，他的双眼中既看到虚拟座舱内外的景色，也看到自己真实的手在搬动一根虚拟的操纵杆。它要求融合在同一个视野中的景象，必须是虚实一致，虚实图像相互对准和不发生矛盾。增强型的虚拟现实技术将是一个具有巨大潜在应用前景的技术。

6. 在个人计算机上实现最简单的虚拟现实环境

上面介绍了一些流行的虚拟现实专用设备和系统结构，事实上，大多数处在研究阶段，成本很高，一般用户经济上根本承受不起。那么，是否就没有机会体验虚拟现实的神奇效果了呢？其实，只要利用普通多媒体个人计算机的性能，人人都能组建一套简易的桌面式虚拟

现实系统。

计算机软硬件的发展速度非常快，普通个人计算机 CPU 的工作频率已经超过了 1GHz，整体的性能达到甚至超过了几年前工作站的性能。由于虚拟现实本身并不限制使用的技术范围，只要能达到目标，可以把各种技术有效地集成为一个成功的虚拟现实系统。因此，对于仅仅想了解虚拟现实技术的用户来说，用普通多媒体计算机完全可以建立一套半沉浸式的虚拟现实系统。

12.3 位置跟踪通道

在虚拟现实环境中，参与者与虚拟环境进行交互，并形成互动。这种互动主要有两种类型：第一种是参与者的操作引起虚拟环境发生变化，可以称为“直接互动”（在虚拟环境意义下）；第二种是参与者的位置发生变化（包括头部、眼球、手势、身体运动等），此时，虽不对虚拟环境操作，但虚拟环境必须随操作者的位置变化而变化（包括场景、声音等），可称为“间接互动”。无论哪一种形式的互动，至少必须让虚拟环境“感知”操作者的位置变化，并对此做出相应的反应，才能使操作者具有沉浸感。这样，对操作者的位置跟踪是必不可少的。

位置跟踪的基础是三维位置传感器。虚拟现实技术中人们所熟知的，无论是头盔显示器，还是数据手套，都依赖于位置传感器，才能成为虚拟现实环境的一部分。迄今为止，关于位置跟踪，头部的平移和旋转、手或手指关节的运动相对说来研究得较多，并取得了较好的进展，而对于其他方面，如眼球运动的跟踪等，难度较大，进展比较缓慢。

本节首先就头部跟踪及手与手势跟踪的有关问题加以介绍。在 12.4 节及 12.5 节中，我们再就视觉与听觉相关的位置跟踪技术进行讨论。本节介绍两类典型的三维位置传感器的原理，即电磁传感器和超声波传感器，同时，对表征虚拟现实技术的数据手套的原理也加以说明。

12.3.1 三维电磁传感器

顾名思义，三维电磁传感器就是利用磁场在空间的传播与检测原理，来感知空间位置的变化，其基本原理如下：此类传感器一般由两部分组成，发送器和接收器。它们均由三个正交线圈组合而成。发送器提供信号源，电流流过正交线圈分别产生磁场，接收器线圈分别感应电流。显然，其感应电流的大小依赖于它与发送器线圈之间的距离与方位。一般说来，发送器的位置是不变的，而接收器则固定在被跟踪的对象上（如头部）（参见图 12.3）。

当然，在使用时，首先应进行标定。例如，为跟踪头部位置，可标定头部正视前方与相应的视景匹配。当头部位姿变化时，接收器的电流发生变化，经检测采样送给计算机，计算机根据电流的变化情况就可计算出当前头部的位姿。基于计算结果，输出给产生视景的计算机，从而产生相适应的视景，并转换成视频信号送给头盔上的两个 LCD 屏幕，从而达到头部位置跟踪的效果。为了避免三个正交线圈之间的相互干扰，必须将三个线圈按时间片激活和检测。

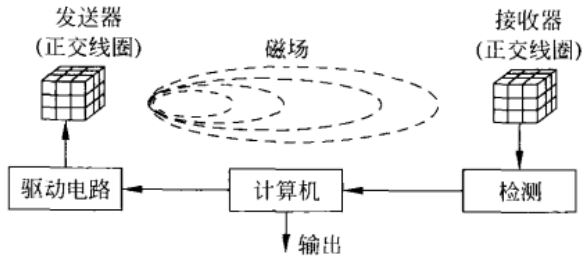


图 12.3 电磁传感器原理图

这类传感器的优点是原理简单，易于实现，包络面大，但其缺点也是显然的，即易受环境中的磁性物质的干扰而降低精度。

典型的三维电磁传感器，如 3-space 的数据采样频率可达 50 点/秒，覆盖范围半径达 150cm 的半球，保证精度的覆盖范围是半径为 75cm 的半球，位置分辨率为 2mm，旋转角度分辨率为 0.35°。

12.3.2 三维超声波传感器

超声波定位在其他领域早已得到了应用。在 VR 系统中，超声波传感器主要用于头部跟踪。下面以 Logitech 公司开发的用于头部跟踪的超声波传感器为例讨论其工作原理(见图 12.4)。

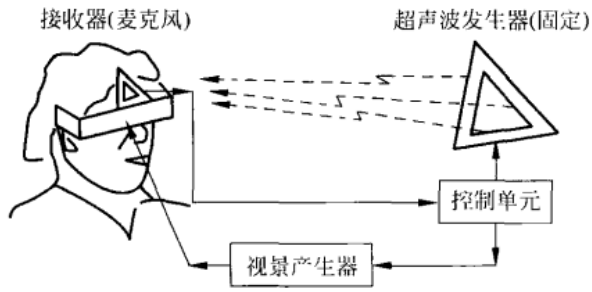


图 12.4 Logitech 三维超声波传感器

三维超声波传感器一般由三部分组成：发送器、接收器及控制部件。Logitech 公司开发的三维超声波发送器由三个分别相距 30cm、紧密固定在一个三脚架上的超声波发生器组成，工作时循环激活。发送器的位置是固定的。接收器由三个安装在三脚架上的微型麦克风组成。系统通过计算声波从发送器到接收器的时间来确定位置和方向。

为了确定三个麦克风所在平面和方向(如果是固定在头盔上，即表示头部位置和方向)，要分别计算每个超声波发生器到每个麦克风之间的距离，这意味着要测 9 个距离。考虑到传递速度与空气的温度有关，实际计算时要根据环境温度调整声速。

控制单元循环激活发送器的超声波发生器，同时对接收声波的麦克风也进行采样。通过计算，将其转换成麦克风所在平面的位置和方向，并送至主计算机。主计算机修改相应的画面并送到头盔显示器中。

该超声波传感器的画面更新速度达 50 次/秒。其主要缺点有：①要求在超声波发生器

与麦克风之间有直射光线。如果有物体挡住了光线或头部偏转太大，则有可能造成信号丢失；②与磁性传感器类似，超声波同样易受金属物体的干扰，甚至其他声波或噪声也会造成干扰。

12.3.3 数据手套(data glove)

数据手套不仅仅是一种位置跟踪设备，实质上它是作为取代传统的键盘、鼠标等实现人机交互的数字化设备。这是 VR 技术突破传统人机交互方式的重大进展。

虚拟现实的目标是消除人及其所处环境和计算机之间的屏障，也就是说，计算机系统及相应的设备向人提供虚拟环境，人可以用其与自然环境交互的各种器官，如眼睛、耳朵、皮肤、手及语言等与虚拟现实环境发生作用。手作为操作、控制的主要器官，如何实现其与虚拟现实环境之间的交互，显然是十分重要的。

虚拟现实技术的发展过程中，三维数字化输入设备引起人们的广泛重视，从而进行了大量研究。基于各种不同虚拟现实环境的要求，技术难度不同、形式各异、价格不等的三维数字化输入设备相继研制出来，典型的有三维鼠标、空间球、操作杆、六维跟踪球等。其中数据手套被认为是最具代表性的。

目前使用得最为广泛的是 VPL 数据手套，它是一种可戴在用户手上的传感装置，能将用户手的姿势转化为计算机数据，与触摸反馈装置相结合，还可具有触摸/力反馈功能。

尽管人手的位姿大约有 30 个自由度，但考虑到在三维环境中，一个物体最多有 6 个自由度(长度、高度、深度、俯仰、转动、偏转)。VPL data glove 具有 6 个自由度的探测器，以监测手的位置和方向，手套背上还装有光导纤维传感器，以监视手指的弯曲，给出各手指关节的角度变化，用于描述手指、大拇指和手腕的相对运动。每个手指的每个关节处均有一圈纤维，连有两个传感器，以测量关节位置(见图 12.5)。

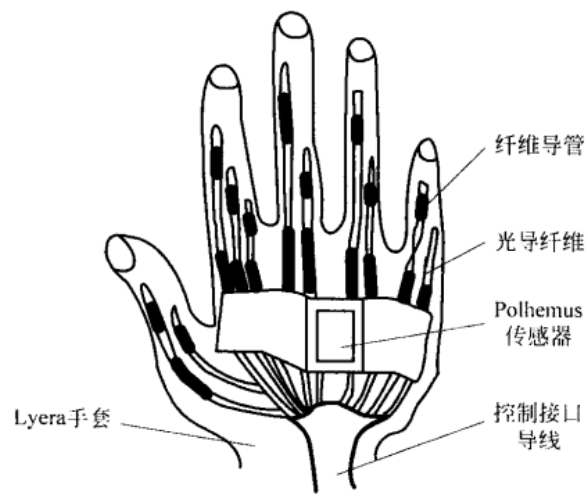


图 12.5 数据手套原理示意图

使用数据手套时，由于每个用户的手大小不一，需要对传感器的初始数据进行校准，而且每次处理结束时均要将其置于新的初始状态，以便实现手及其关节的位姿的数据转换。

12.4 视觉通道

虚拟现实系统之所以使用户具有“沉浸”感，很大程度上依赖于虚拟环境的逼真程度，而人类对周围环境信息的感知 80%是通过眼睛获取的，因而，虚拟环境的质量主要取决于视景生成的好坏。从 VR 的研究历史中也可以看到，在关于虚拟环境系统人机接口部分的所有设备中，视觉显示器得到最多的关注，大量的研究工作致力于开发合成环境系统的专用视觉显示器。

12.4.1 视景的生成过程

视景中的图像是由计算机根据环境的需要，利用给定的条件与模型，对图形数据和图像数据进行计算后，通过视景通道生成的，其基本过程如图 12.6 所示。

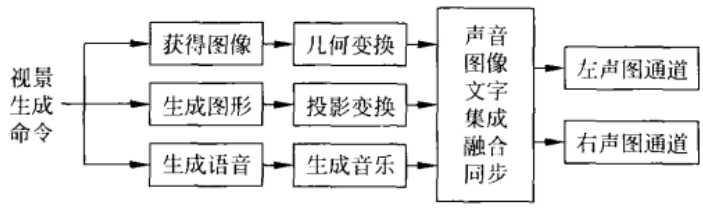


图 12.6 视景的生成

在图 12.6 中，声音也列入到视景生成中，这主要考虑声音和视景往往是相关的。人类的视觉是最灵敏的，VR 要求产生大视野、双眼立体显示。常见的视觉显示器是头盔式显示器(HMD)。计算机在屏幕上产生的图形与图像实际是二维的。如何才能使人获得立体感呢？这就需要了解人是如何从视觉信息中获得立体信息的。

人是通过眼睛来感知图像信息的。如何由这些图像信息获得深度感，这是一个复杂的生理过程。尽管这一过程至今尚未得到全面的解答，但“双目视差”被认为是造成立体感的重要方面。双目视差是指当人们用双眼看一物体时，由于左右两眼的位置细微差别，所看到的景象不完全一致。人的大脑则自动将这两幅不完全相同的图像融合起来并获得立体信息。目前，大多数 VR 系统的视觉通道均是基于这一原理来实现的。

12.4.2 基于投影变换原理三维立体图的生成算法

基于计算机图形学原理，首先生成对象的线框图，然后在线框图上加上颜色、纹理、阴影、光照以及其他物理特性，从而生成一幅逼真的图像。本节介绍投影变换的基本原理。

(1) 单视点透视投影

计算机图形学中，常用的一种三维图形显示处理技术是标准投影变换，它采用单个视点进行投影，简称为单视点投影，其基本原理是：取左手坐标系，以视点为投影中心，三维物体上的点投影到某一投影平面上，便在该平面内产生三维物体的像。

对该三维物体的某一个点来说，透视投影的任务在于如何求得该点到投影中心的连线与投影平面的交点(即投影点)(参见图 12.7)。

设视点到投影平面的距离为 d，其坐标为 $Cop(0,0,-d)$，将三维空间物体上的点 $p(x_p,y_p,z_p)$投影到 x-y 平面上的点为 $p_s(x_s,y_s)$，则有

$$x_s = x_p \times d/(z_p + d) \tag{12.1}$$

$$y_s = y_p \times d/(z_p + d) \tag{12.2}$$

(2) 左、右双视点透视投影算法

为了适应双目视差立体成像原理的需要，左右双眼都应该进行透视投影，这时存在两个投影中心。仍采用左手坐标系，投影平面位于 $z=0$ 处，两眼之间的距离为 e，左眼的投影中心记为 $L_{cop}(-e/2,0,-d)$，右眼的投影中心记为 $R_{cop}(e/2,0,-d)$(参见图 12.8)，其中 d 为两视点连线到投影平面的距离。物体表面上的任意一点 $p(x_p,y_p,z_p)$分别以左眼投影中心和右眼投影中心向投影平面进行透视投影，则左视点的投影计算公式为

$$x_{sl} = (x_p - z_p \times e/2)(d + z_p) \tag{12.3}$$

$$y_{sl} = y_p \times d/(d + z_p) \tag{12.4}$$

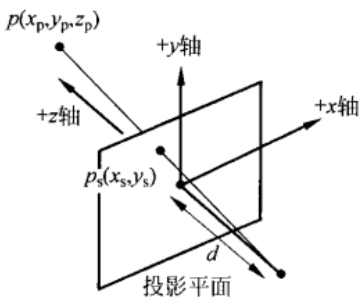


图 12.7 单视点透视投影

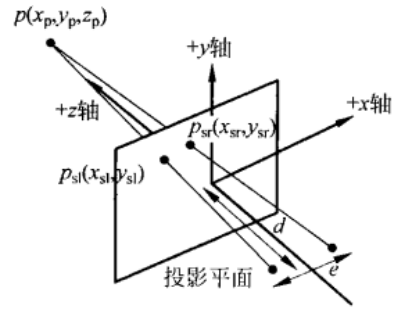


图 12.8 双视点透视投影

同样，对右视点，其投影计算公式为

$$x_{sr} = (x_p + z_p \times e/2)(d + z_p) \tag{12.5}$$

$$y_{sr} = y_p \times d/(d + z_p) \tag{12.6}$$

其中(x_{sl},y_{sl})为 $p(x_p,y_p,z_p)$按左视点投影在 x-y 平面上的点，(x_{sr},y_{sr})为 $p(x_p,y_p,z_p)$按右视点投影在 x-y 平面上的点。

(3) 基于相关性原理的立体图形光线跟踪算法

实际上，如果我们已经计算得到左视点投影的点，则可由其推出右视点投影的点，即

$$x_{sr} = x_{sl} + e\frac{z_p}{d + z_p} \tag{12.7}$$

而 y 坐标的值是相同的。根据这一相关性，Ezell 和 Hodges 提出了一种根据左眼的光线跟踪图像按式(12.7)推导出右眼视图的方法，后来，Adelson 和 Hodges 对上述方法进行了改进。

当按照式(12.7)推导右眼视图的时候，会出现以下三种难于确定的像素，即重像素、坏像素及像素丢失。下面分别讨论解决这些问题的方法。

① 重像素的去除

重像素是指有多个像素被投影到一个像素点上。这时必须解决使用哪一个像素值的问题。下面首先分析重像素是如何产生的。

考虑到双目投影计算 y 的坐标不变，我们可在 x-z 平面讨论重像素问题。设在右视点 R_{cop} 发出一光线(参见图 12.9)，通过 $PIX(x_1,y,0)$，它与空间物体交于 z，则 z 的方程为

$$z = -d + d \times T = d \times (T-1)$$

其中 T 是一大于 0 的数。这意味着，如果空间有若干个物体，凡与该光线相交的表面上的点从右视点来看，其透视投影点均落在 $PIX(x_1,y,0)$ 点上。问题是在图像上如何确定哪一个是最近的点。考虑按 y 坐标一行一行地自左向右同时计算两幅视图，先计算左视图，相当于从 L_{cop} 发出光线与空间物体表面相交，在投影平面的投影点为 $PIX(x_2,y,0)$，当 z 值相同时，可得到如下计算式

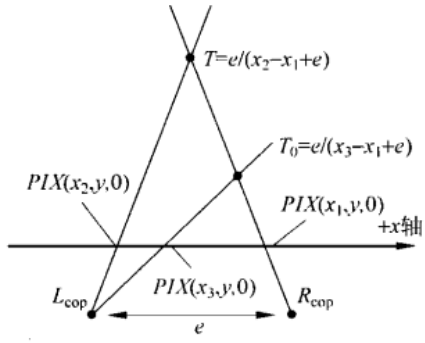


图 12.9 重像素问题

$$x_1 = -e/2 + T \times (x_2 + e/2) \tag{12.8}$$

$$x_2 = -e/2 + T \times (x_1 + e/2) \tag{12.9}$$

$$T = e/(x_2 - x_1 + e) \tag{12.10}$$

假设有一空间物体表面上的点 T_0，对右视点来说，其在该投影平面透视投影点也是落在 $PIX(x_1,y,0)$ 点上，但从左视点来看，在该投影平面透视投影点为 $PIX(x_3,y,0)$，此时，可得到

$$T_0 = e/(x_3 - x_1 + e) \tag{12.11}$$

因为 $x_2 < x_3$，所以可以判断，$T_0 < T$，从右视点来说，T_0 点位于 T 点的前面，显然应保留 T_0 点的投影像素值。因此，解决重像素保留问题只需要判断 T 值的大小就行了。实际上，可以在程序设计时，分别对 z 轴与 y 轴按自大到小进行扫描，在处理的同时保存其相应的投影像素值，就能使右视图的投影像素不会因像素重叠而出现显示错误。

② 坏像素去除

坏像素是指当从左视图向右视图投影时，原来在左视图中相邻的两个像素在重投影后的右视图之间出现一个 gap(长度为 1 到 $e-1$)，这样，就可能在该 gap 中出现别的物体的全投影像素。

我们用图 12.10 说明这一现象，图中物体 A，B，C 分别对左视点和右视点在 x-y 平面进行投影。

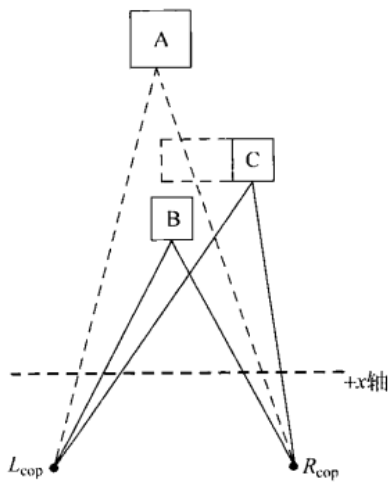


图 12.10 坏像素的产生

物体 B 与物体 C 相对于左视点在 x-y 平面的投影像素是相邻的，但是对右视点则相隔了一个 gap，这样，A 物体就应投影到该区域中。这是正常情况。然而，如果物体 C 比较大(如虚线所示)，则物体 A 相对于右视点的投影就应该被遮挡而变成不可见。在这种情况下，就应该清除在该 gap 区间上的投影像素。

③ 像素丢失问题

在重投影后，右视图中有些像素没有值，这需要通过对这些像素进行光线跟踪计算来解决。

关于视景生成还有其他一些方法，特别是，快速全信息图像成像算法是近几年来迅速发展起来的卓有成效的技术，其速度可比传统的方法快 100 倍以上。但其原理需要花较多的篇幅才能比较清楚地加以说明，有兴趣的读者可参考有关文献。

12.4.3 典型的立体视觉设备

立体视觉设备是 VR 技术研究最多、发展最快的方面。如果根据其使用方式加以分类，大致可分为三大类：头盔式显示器、立体眼镜和投影式显示器，其中最典型的是头盔式显示器，尽管它也包括显示听觉信号的装置(耳机)及测定头部位置和方向的装置(头部跟踪器)，一般仍将其归类为立体视觉设备。这里，仅就头盔式显示器的原理及功能作一简单介绍。

头盔式显示器(head-mounted display，HMD)是专为用户提供虚拟现实中景物立体图像的设备，其视觉通道用两个 LCD 或 CRT 显示器分别按一定频率向两只眼睛提供图像信息，图像信息来自计算机，两幅图像存在着细微差别，类似于双眼视差获得的图像。

计算机输入图像有两种方式：一种是两幅图像分别由两台计算机发送，驱动左、右眼的两个显示器，其同步问题依靠控制两台计算机来实现；另一种是采用一台计算机分时交替生成用于驱动左、右眼的两个显示器的图像，此时的同步则依靠应用软件来实现。

作为立体视觉设备，头盔式显示器由三部分构成：显示部件、光学部件及头盔(参见图 12.11)。



图 12.11 头盔式显示器

显示部件的作用是把计算机输出的图像显示出来，从实现技术来说，目前主要有三种：液晶显示器(LCD)、阴极射线管(CRT)和彩色开关显示器。前两种显示器已广泛使用在计算机、电视机、监视器等设备中，彩色开关显示器是将这两者结合起来产生的一种新型显示设备。它采用传统的单色显示器来产生图像，液晶显示器(LCD)设备放在其后用来选择显示颜色。从目前使用的情况来看，液晶显示器是最普遍的。为了减少头盔式显示器的体积，显示屏与观察者的眼睛之间的距离一般只有几厘米，显然，计算机产生的图像不能直接在显示屏上成像，否则，单靠眼睛在这么近的距离下是无法看清图像的。

光学部件就是为了解决这一问题的。光学镜头将屏幕产生的图像成像在观察者能看清的距离，使虚拟环境中的物体看起来的尺寸与真实尺寸差不多，并使图像覆盖尽可能大的视场，从而大大增强了沉浸感。当然，据测试，人的眼睛水平视场角可达180°，加上眼球转动，在头部不转动的情况下，水平视场角可达270°，而目前的光学部件提供的视场角大约为90°，似乎相差甚远。实际经验表明，在具有头部位置跟踪功能的情况下，90°视场角产生的沉浸感足够令人满意了。

为了将显示部件与光学部件结合起来，并便于用户使用方便，头盔既是一个定位部件，也是一个功能部件。它将显示功能、头部位置跟踪功能、立体音响功能集成在一起，达到用户视、听的沉浸与交互。头盔显示器的性能主要包括以下几个方面：

① 视场角度

人的单眼水平视场角为150°，垂直视场角为135°，两只眼睛总水平视场角为180°(不考虑眼球运动)，这样视场重叠角为120°。头盔显示器应尽可能接近这一指标。

② 分辨率

人眼的最高分辨率为 1 弧分左右，目前的显示技术及计算机成像技术均不能达到这一水平。例如，3.43cm 的 LCD，目前世界上最高水平的点阵为 480×360，按120°的水平视场角计算，其分辨率为 15 弧分。为此，在实际实现时，往往通过提高中心分辨率、牺牲边缘分辨率的办法达到相对较好的效果。

③ 延时

延时主要由于两方面原因：计算机成像延时和显示器显示延时。如前所述，总延时不应超过 0.1s。因此在设计 VR 系统时，要综合考虑计算机的性能、显示器的性能，以及显示对象模型的复杂程度。

④ 图像质量

图像质量受多方面因素的影响，如色彩的丰富程度，光学镜头的畸变程度、聚焦与汇聚能力等。

⑤ 尺寸与重量

显然，头盔应越轻越好，以减轻用户的不舒服感。尺寸的适应性也很重要，因为人的头部尺寸存在差异、眼睛瞳孔之间的距离存在差异、是否配带眼镜等因素都会影响到使用的效果。

12.5 听觉通道

有声电影较无声电影更具有存在感和临场感是众所周知的事实。例如，当视觉通道显示出一列火车由远而近驶来，如果伴随由小变大的车轮与车轨碰撞声、鸣笛声，以及与空气摩擦而发出的呼啸声，其真实性效果是无声音时无法比拟的。据统计，人类从外界获取的信息中约有 15%来自听觉通道，因此，为获得良好的沉浸感，一个与视觉通道同步的听觉通道是必不可少的，听觉通道是虚拟现实环境的另一重要组成部分。

在虚拟现实环境中，与听觉通道有关的内容主要包括虚拟声音的产生与定位，考虑到人与虚拟现实环境的交互，语音的识别与合成技术也是一个重要的方面。

12.5.1 虚拟声音的产生

当今的音响合成器和录音技术已达到很高的水平，产生、编辑以及通过计算机播放“完美”的声音技术已经得到广泛的应用。虚拟现实环境的虚拟声音与一般的音响合成和录音技术的区别在于：虚拟声音必须与视觉通道同步。在虚拟现实环境中，虚拟声音可分为两大类：语音声音与非语音声音。

对非语音类声音，如振动声、摩擦声、干扰噪声等，可以根据物理学原理来产生虚拟声音。例如，对于振动声可采取频谱组合法，根据对象的类型及与其交互作用的环境建立声音

世界模型库，并将每一个声音模型与之联系起来。另一种方法是将录制的声音与产生该声音的物体交互作用联系起来，通过声音事件文件来描述声音世界。这种方法的缺点是：如果物体交互作用的物理学原理不知，甚至到底产生什么样的声音也不知道，就无法产生相应的声音。例如，假设要产生木片虚拟地插进铁块中的虚拟声音就无法实现。

对语音类声音，虚拟声音包括语音的识别与合成。这是虚拟现实技术智能化的典型特征，也是最为困难的课题。为了完整起见，本节对语音的识别也做一简单介绍。这实际上是要求计算机具有听(理解)与说的能力，即感知、理解人的语音，并能以人可理解的声音传递信息。

语音识别与合成的研究有五六十年的历史了，早期的研究大都是以“词”为对象。在用计算机进行语音识别的研究中，Denes 与 Mathews 提出了“时间规整”这一重要概念，一度成为语音识别的主流方法，其基本思想是：将“标准”的语音模型与待识别的语音相比较，以时间为轴记录语音信号，将待识别的语音特征映射到语音模型，如果两者特征对正时是线性关系，则意味两者匹配，从而实现其识别。实际上，由于不同的人对同一词的发音不完全相同，即使同一人在不同时间对同一词的发音也不完全相同，一般来说，难以得到线性的映射关系，待识别的语音信号的时间轴往往要不均匀地扭曲或弯折。时间规整的目标就是从已有的语音模型库中找到一个最优匹配语音模型。

例如，图 12.12(a)及图 12.12(b)中，设 A 为某一单词的标准语音信号，B、C 是待匹配的信号。图 12.12(a)的语音信号 B 与 A 之间的映射函数是一条曲线，因而 X 是非线性映射；而图 12.12(b)中的 Y 基本上是直线，因而是线性映射。完全是直线映射的情况是少见的，而大多数情况则是不同程度的非线性映射。非线性映射情况下进行语音识别的关键是寻找规整函数。动态时间规整(dynamic time warping，DTW)的规整函数定义为：设“标准”语音模型特征序列为 $A=a_1,a_2,\cdots,a_M$，待识别的输入语音的特征序列为 $B=b_1,b_2,\cdots,b_N$，并且 $M\neq N$，寻找函数 $m=w(n)$，使得输入语音的特征序列的时间轴 n 非线性地映射到“标准”语音模型特征序列的时间轴 m，并满足

$$D=\min_{w(n)}\sum_{n=1}^{N}d(n,w(n)) \tag{12.12}$$

式中，$d(n,w(n))$是第 n 帧的输入矢量与第 m 帧的模型矢量之间的距离，D 是相当于最优时间函数下两者的距离测度，用于对输入词评价“打分”的依据。如果某一输入语音信号与模型库中的某一标准语音信号距离测度最小，则可认为两者匹配，从而达到识别的目的。

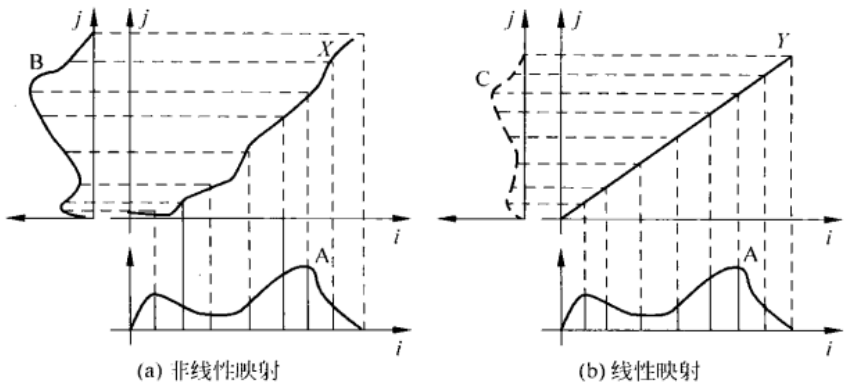


图 12.12 时间规整

虚拟语音合成技术有两种途径，一种是将可能出现的语音信号事先存储在计算机中，根据场景的变化，在需要时，选择合适的语音信号在声音通道输出。这种方法本质上是语音还原技术，声调、语调不能控制，而且需要大量的存储容量。第二种是真正的语音合成技术，目前主要研究的是按规则的文本-语音转换。转换的方式主要有三类，即波形编辑、参数合成、基于规则合成。下面就这三类方式分别加以说明。

波形编辑方式：任何一句话都可以将其分解为单词、词组或短句。波形编辑方式将单词、词组或短句作为“基元”，并以波形参数形式存储。语音合成时，将波形参数编辑起来，语音合成器将其连接起来形成语句进行输出。用这种方式合成的语句的优点是可懂度强，单元语音的自然度好，缺点是音调、语调能控性差，听起来觉得慢。

参数合成方式：以字、词作为基元，利用语音生成模型，提取模型参数，然后将这些参数存储起来。语音合成时，合成器将这些模型的参数连接起来，合成所需要的句子，最常用的是共振峰合成与线性预测合成(LPC)。

基于规则合成：按照语音规则将预先存储的语音单元连接起来，产生可懂的连续语句，因此称为基于规则合成。这种方法的要点是如何确定语音规则。一般语音基元比较小，如音节、音素、二分音、半音节等，这样语音单元数目少，便于组合，但合成的规则多。反之，如果语音单元较长，合成的规则数目少，但语音单元的数目则增多。

12.5.2 虚拟声音的定位

在一般情况下，虚拟现实环境的声音是“虚拟声音”。Currtll 对虚拟声音进行过定义，即：“虚拟声音是一种已记录下的声音过程，包含明显的音质信息，能改变人的感觉，让你相信这种记录声音正在实实在在地产生在真实的世界中。”虚拟声音与人们熟悉的立体声有本质的区别。美国航空航天局(NASA)的实验表面，如果采用通常的立体声技术制作声源，实验者戴上立体声耳机，感觉到声音在头颅内部回响，而不是来自外界。必须同时改变声音的混响时间差和混响压力差，才会感觉到声源位置的变化。

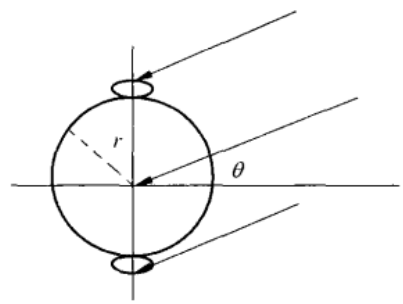


图 12.13 混响时间差

混响时间差(interaural time difference，ITD)是指声波分别传播到左右耳的时间差，研究表明，混响时间差与声音的频率 f、声源位置(以声音方向与两耳连线之垂线夹角 θ 表示，参见图 12.13)、头部半径 r 以及声音的速度 v 有关

$$\mathrm{ITD} = \frac{f \times r \times \sin\theta}{v} \tag{12.13}$$

其中，当频率 f 低于 4 000Hz 时，f 取值为 300，而当 f 高于 4 000 时，取值为 3 000。例如，设声音速度为 344m/s，头部半径为 0.09m，按声音频率低于 4 000Hz 计算，当声源来自正后方，θ 为 0°，按顺时针方向计算，左耳为90°，右耳为270°，ITD 的值为 0；如果 θ 为45°，可得 ITD 为 0.005 566s，即混响时间差为 56.66ms，这就要求在虚拟环境中右耳扬声器发出的声音早于左耳扬声器 56.66ms。

混响压力差(interaural pressure difference,IPD)是指声波对左右耳作用的压强差。在声波传播过程中,如果声源离左右耳的距离不同,则存在混响压力差,有时也称为"头部阴影效应"。研究表明,混响压力差与声音的频率 f、声源位置(以声音方向与两耳连线之垂线夹角 θ 表示)有关,可用如下公式表示

$$\mathrm{IPD} = 1.0 + (f/1\,000)^{0.8} \times \sin\theta \tag{12.14}$$

f 的单位为 kHz。实验表明,在低频时,混响压力差很小,当频率达到 3 000Hz 时,混响压力差显著增大,成为声源定位的主要信息之一。

当然,最新的研究结果表明,仅仅考虑混响时间差和混响压力差仍会产生某些不足,也就是说,即使将立体声耳机中的声音改变混响时间差和混响压力差,在某些情况下仍觉得声音不是来自外部,自然性仍然不够,说明人类的听觉功能是一个十分复杂的过程,于是,人们力图研究与头部有关的听觉功能,并用传递函数的方式进行描述(head-related transfer function,HRTF)。尽管如此,混响时间差和混响压力差仍然是声音定位的主要因素而加以采用。

12.6 触觉、力反馈

如 12.2 节所述,虚拟现实技术在触觉/力反馈方面的进展是最为缓慢的,尽管人们进行了大量的努力。虚拟环境中的人机交互,触觉/力反馈通道也是非常重要的一方面,实验表明,简单的触觉/力反馈与视觉结合比只使用视觉的效果好 30%以上。

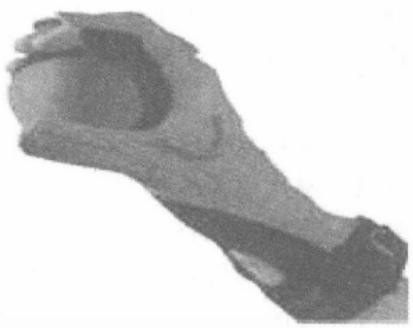


图 12.14 触觉及力反馈示意图

触觉/力反馈通道是指那些可以实现与虚拟环境或远程遥作系统进行手工交互的设备(参见图 12.14)。它们用来完成现实世界中通常是用手来进行的活动,比如物体的手工探查与操作。一般来讲,它们接受来自用户的动作命令同时回显出适当的触觉图像。这种触觉上的交互可以伴随着诸如视觉、听觉等其他感官方式的仿真。计算机键盘、鼠标、空间球以及对游戏者的施力作出反应的游戏杆是相对简单的触觉/力反馈通道,数据手套和数据外衣是比较复杂的触觉/力反馈通道。

与视觉和听觉的纯感知特性不同,触觉系统既可以感知环境,同时也可以对环境产生作用。人手是个多功能的器官,可以完成多种动作,如对物体的挤压、抓握、扭转、击打等。它既可以探测出物体的表面纹理、形状和柔软度等特性,也可以拿起工具来修理一块手表或者敲碎混凝土。在视听之外,能够触摸、感觉并操作环境中的物体才可以给人一种真正融入周围环境的真实感受,否则这种真实感受是不充分的。因此,即使是实现视听显示与简单触觉接口的同步操作,也要比单独大幅度地增强视觉上的逼真程度更能有效地提高虚拟环境的融合感。倘若真实环境或虚拟环境不能让人触摸和感觉物体,那么这种环境一定是十分枯燥的,并且严重地阻碍人的交互能力。

本节首先对人体触觉系统进行简单讨论,然后介绍目前的触觉/力反馈通道技术,最后

对这一领域的技术给出综合性的结论。

12.6.1 人体触觉系统

为开发触觉/力反馈通道，理解人体自身触觉系统中各子系统所起的作用是十分必要的。这些子系统包括机械、感觉、运动和认知子系统。

人手的机械结构是由错综排列的19块骨头组成，通过许多几乎无摩擦的关节相连，外面覆盖着软组织与皮肤。通过数量众多的腱，这些骨头被附着在内部、外部的20条肌肉上，以支持人手所具备的22个自由度的活动功能。感觉系统包括大量的、各种类型的存在于皮肤、关节、腱与肌肉之中的感受器与神经末梢。适当的刺激会激励这些感受器，使它们通过传入神经向中枢神经网络发出电脉冲，继而中枢神经通过传出神经向相应的器官发出指令以执行合适的动作。

通过手与物体接触而获得的感觉信息可以分为两类：①触觉信息。是指与物体相接触的感觉，主要是通过低阈值的机械感受器对接触区域的皮肤产生神经刺激而产生的。②运动觉信息。是指与力相伴的肢体位置与运动的感觉，主要是由位于关节周围的皮肤、关节囊、腱、肌肉内的感受器来传递的。当物体接触被动的、相对静止的手时，除了以前的有关肢体姿势的运动觉信息以外，只有触觉信息被传输；当手在做主动的、自由运动时(也就是与物体或皮肤的其他部分无接触)，只有运动觉信息被传输。但是触觉信息的空缺也就相当于传输了该运动是自由的这一信息。所有的感觉和操作任务都涉及了运动觉和触觉两类信息的传输。另外，自由神经末梢和一些感受温度与痛觉的特殊感受器也参与到了这些任务中。

对于成功地完成任务而言，接触条件的控制与感受这些条件具有同等重要的地位。人类可以控制从快到慢很大范围内的动作。在用两个手指(拇指和食指)捏住物体向上提升的试验中，当物体开始相对滑离手指肚时，对物体的抓力会在70ms内开始增加。来自于皮肤传入神经的感知信号对于这一动作起着重要作用。很明显，皮肤和皮下组织的机械特性，连续监视任务的大量传感器所提供的丰富的传感信息，以及这些信息与运动系统动作的相互协调作用，使人能够完成对物体的抓取操作。

人体触觉系统可以感知和区别触摸对象的细微差别，然而，触觉/力反馈过程可以分解成两个基本步骤：探测和操作。“探测”主要关注于对物体属性的提炼，它主要包括对物体表面属性(如形状、纹理)和体积属性(如质量、柔度)的分析或鉴别。“操作”主要是个动作任务，大致可以分为两类：精确任务(如修表)和力量任务(如使用锤子)。

在任何涉及与物体发生实际接触的任务中，无论它是探测还是操作，皮肤和皮下组织的生理特性对于成功地完成任务都起着重要作用。举例来说，在几乎所有的精确任务中，我们都要用到手指肚(finger pad)，它是由表面无毛的绷紧的皮肤包裹着处于半液态状的脂肪软组织构成的，厚约1mm。作为一块肉体组织，手指肚展示了复杂的机械行为——异构性，各向异性，速度和时间的相关性。皮肤的柔软性和有摩擦性，以及手的感知与运动能力，使人既可以一直保持接触地滑过物体表面去探测表面形状，也可以牢牢地抓住物体进行操作。皮肤的机械负荷，通过皮肤的机械信号的发送，以及皮肤机械传感器的传感很大程度上都依赖于皮肤和皮下组织的机械特性。

当我们用手接触或挤压物体时，触觉系统和运动觉系统都对这一接触力产生感知。总的来看，接触力可能是最重要的变量，它决定了感觉系统的神经信号并通过动作控制接触条件。接触力的阈值在很宽的条件下都是所受力的5%～15%，在与之相关的人对不同重量物体的分辨能力的实验研究中，高于阈值10%的量就已经被观察到了。在19世纪末Weber观察到的一个有趣的现象是冷的物体感觉比同等重量的暖物体要重。在用两个手指捏住物体上下移动的实验中，Johansson和Westling证明了在物体的提升中抓和举的力的比例是受到精确控制的，这样物体才不会滑落。当使用局部麻醉使感觉信息受到阻碍时，这种精确控制的能力就大大降低。因为人无法再感受诸如下滑的发生等接触条件并同时采取相应的补偿来增加抓力。所以，在接触活动中既需要力的感知，也需要利用这些力来控制接触条件。

研究表明，对触觉感知系统来说，人体通过触觉感可以辨别出0～1kHz的振动。手指的单点空间分辨率为0.15mm，双点的阈值为1mm。人对光滑平板的检测阈值为2μs高的单点和0.06μs高的栅格。而运动觉感知系统，人体可检测到持续1s以上的小于1°的关节旋转。运动觉系统的带宽为20～30Hz，指关节的阈值是2.5°，腰与肘关节的阈值是2°，肩关节的阈值是0.8°。人体运动系统中，四肢运动的带宽是作用方式的函数，对不可测信号为1～2Hz，对周期信号为2～5Hz，对反射运动为10Hz，对位置和运动的不同阈值已精确测量至8%。三个系统共同参与的主动触摸，对于10mm的长度，检测阈值是1mm；对于80mm的长度，检测阈值则增加到2.4mm。接触力的阈值是参考力值的5%～15%，手指可实现的最大可控制力约为100N，分辨率为0.04N或1N甚至更大。

当然，人体的触摸过程是一个十分复杂的运动、受力、感觉、反应、控制的过程，这一过程的生理学本质，迄今为止，人们的认识还十分肤浅，有待进一步深入研究。这也是触觉/力反馈通道技术目前还相对落后的根本原因。

12.6.2 触觉/力反馈技术

如前所述，触觉/力反馈通道是由机械组件构成的与人体产生物理接触并与人的神经系统进行信息交换的一类设备，在使用触觉/力反馈通道完成某项活动时，使用者通过操作触觉/力反馈通道传输所做出的物理动作，然后，触觉/力反馈通道通过仿真使用者的触觉和运动觉系统来显示触觉感知信息。因此，通常意义下触觉接口有两个基本功能：测量手的位置和接触力；显示接触力和位置。在这些位置和接触力变量中，选择哪些作为运动变量(输入到计算机中去)，哪些作为显示变量(输入于人)要取决于硬件和软件的设计以及该接口是用来完成何种任务，因此，可以借用机械学和机器人学中的术语来描述接口的各个子系统。

1. 触觉/力反馈接口技术分类

触觉/力反馈接口技术依赖于触觉交互的类型，可以从不同角度对触觉交互的类型加以分类。

从受力的角度，触觉交互基本可分为三类：自由运动，即与环境中的物体不产生物理接触；由不平衡合力形成的接触，比如用手指压一个物体；由自平衡力形成的接触，比如，抓握并挤压一个物体。抓住一个物体并把它由一个位置移动到另一个位置就包括了以上三类

要素。

从交互的工具角度，触觉交互很重要的区别在于我们是直接触摸、感觉和操作物体还是借助于工具来进行。这两类不同的交互方式直接影响接口设计的复杂性。两者共同点是触觉接口都能够测量使用者手的位置和姿态，显示力的作用。不同的地方在于直接触摸较借助于工具具有更大的复杂性，包括受力方式、触摸感受等。

从力的显示方式分类，触觉接口可分成基于地面的（比如游戏杆和其他手动控制器）和基于身体的（比如数据手套和数据外衣）。如果这些力是自平衡的，接口就无须有机械基点，但若是不平衡的，接口就需要固定在某个地方。连接在地板上的游戏杆就是基于地面的显示装置，附着在使用者前臂上的外廓装置就是基于身体的显示设备。基点的选择影响了使用者在接触虚拟物体时感受的重点。

2. 触觉/力反馈技术发展

我们分别从硬件和软件两个方面讨论触觉/力反馈通道技术。在硬件方面，主要包括力和力矩的显示装置；触觉显示装置。

（1）硬件

游戏杆是这些技术中最早开发的，最初是用来控制飞行器，通过机械连接与飞行器表面相连，表征作用于飞行器表面的力的负载。游戏杆有被动型的（无力的反应），比如用于光标定位的游戏杆；也有主动型的（有力的反应），比如当今许多的飞行控制杆等。Measurement System 公司已经在市场上推出了几种两个自由度和三个自由度的位置感知游戏杆，其中有些能够感知但不能显示。AT&T 公司的迷你游戏杆的带宽较高，两个自由度，并有力的反应功能。

目前许多带力反应功能的手控器用于远程操作控制，最多有 6 个自由度。北卡罗来纳大学的力反应手控器 argonne 机械手臂被成功地应用于分子对撞仿真和扫描隧道显微镜的数据交互中。“MIT 砂纸”能够显示出虚拟材质的三维游戏杆。建在 MIT 人工智能实验室中的 PHANTOM 则是能够传递虚拟物体感觉的多连接低惯性设备。哥伦比亚大学利用磁悬浮技术开发出了高性能的手控器。

触觉/力反馈的一个关键技术是表面压力反馈技术。虚拟现实环境中的表面压力反馈可分为两种类型：①通过工具与虚拟物体接触；②用户手的表面（皮肤）与虚拟物体接触。无论哪种情况，用户实际上都不可能像抓住真实物体那样抓住虚拟物体，因此，必须要有能向用户提供触摸效果的设备，其中的核心问题是如何测量施加到虚拟物体表面交叉段（cross section）的力。

如果交叉段发生在工具表面上，这是“间接接触”。由于工具至多可考虑 6 个自由度（人手指大约有 20 个以上的自由度），形状一般较手指简单得多，且没有柔韧性，因此较之第二种情况，表面压力的检测与反馈要简单得多。这种测量接触表面作用力，并把信息反馈给被计算物体的模拟过程，称为表面显示。

基于表面显示概念和技术，日本东京大学的广濑等人研究出了一系列表面压力反馈装置。最新的装置分为机械部分和控制部分。机械部分又分为跟踪和显示两部分，跟踪部分为二自由度，目前正在设计六自由度的跟踪设备。控制系统采用 PC，并通过 TCP/IP 与主机相连。PC 与主机协同工作，向用户提供物体表面，计算周期为 100ms。在哈佛大学，

HOWE 已开发出一种两指控制的遥作系统，它可以用拇指与食指的抓握来执行精确操作。当今最为复杂的一套力反应设备当属由 Sarcos 公司与犹他州大学的工程设计中心及 NOSC 共同设计的 dextrous teleoperation system master。尽管它基本上是基于地面的，但是通过与使用者前臂和上臂的连接点，它仍具有活动空间大等优点。这一设备利用高质量的液压执行器，在相对高的带宽下提供了较大动态范围的力的施加作用。另一个系统的例子是 Huter 等人实现的双手遥作的微型机器人。它可以达到宽带宽和高精度的双重需求。这些设备的改进装置已被应用于眼外科手术，并代表了当今技术所能实现的最高性能。



图 12.15 数据手套的触觉及力反馈示意图

力学反馈手套是一种虚拟手控制系统(参见图 12.15)，在计算机的控制下，通过手指的运动，直接或间接(通过操作器)操作虚拟对象，手可感知触觉力和力反馈。

力学反馈手套最早用于遥控机器人。英国 Salford 智能机器人研究中心开发了一个手控制器触觉反馈系统，与机器人相连。该力反馈手套原理十分简单(参见图 12.16)，采用了两个手套，第一个为输入手套，装有 20 个压敏元件，当戴上手套时，随着手用力，用户就可感觉到压敏元件的阻力，另一只为输出手套，在对应位置有 20 个空气囊，它们受 20 个空气泵控制。压敏元件的输出经模数转换输入计算机中，计算机处理后再经数模转换输出到控制空气泵，并可根据压力大小对空气囊充气以改变气囊的压力，从而就有压力感。Advanced Robotics Research 公司采用这种方法设计了“teleact corore”气动式触摸反馈设备，后来与数据手套相结合，取消了力量感应器。比较新颖的力反馈手套是 RM-Ⅱ，多个手指都有位置感应和力的反馈。

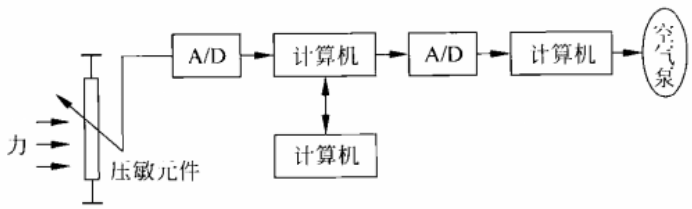


图 12.16 力学反馈手套原理

通常，触觉/力反馈通道接收使用者的动作命令，然后向用户显示适当的触觉图像。触觉图像包括力和位移的显示，以仿真观察者在 VR 中与虚拟物体的机械交互作用。一般来讲，这些图像刺激观察者的触觉和运动觉通道并由观察者的动作所驱动。主要的传递信息有接触物体的模式、物体的机械属性及 VE 中探测和操作所涉及的运动和力等。

虽然目前力显示设备可用于近似地与虚拟物体的交互，但是，触觉信息的显示需要对接触区域更精细地显示。形变显示通过控制在皮肤上的变形或力的分布来传递局部接触形变，这一功能已经被采用直流螺线管、形状记忆合金及压缩空气作用的激励器阵列所实现。Monkman 于 1992 年提出了使用电动黏性流体作用于连续表面的方法。振动感显示装置通过接触皮肤的振动测针阵列来传递机械能，Opticon 和 Bagej 公司的触觉刺激器均属于这一类。John Hopkins 大学正在开发一种特别的桌面触觉阵列，可以实现高质量的形状改变和

零到几百赫兹频率的振动功能。

(2) 软件

产生触觉图像的软件可以被划分为三大类：触觉交互软件、虚拟物体与环境的物理模型软件、润色触觉图像的软件。

触觉交互软件主要是读入触觉接口设备的状态，比如在读入位置或力传感器的信号时所必需的信号调整与抑制噪声的软件就属于这一类。在用于跟踪手势的外廓类设备中，也需要基于人手运动学模型的高水平软件来解释与手势相对应的传感器信号。

虚拟物体与环境的物理模型软件大多可以为每一种模态建立单独的模型并通过协调算法来保持各模态间的一致性。对于触觉交互方法所需的计算量相当大，即使是采用超级计算机也很难做到实时仿真。对多模态图像的模型简化是十分必要的，这不仅由于计算上的困难，也还受到目前显示设备的能力所限。因此，尽管物体的视觉、触觉等感官都是同一物理机制，但对于每一模态却需要不同的近似。比如，视觉图像是物体的二维投影标量，但触觉图像则一般是三维向量。对于逼真的视觉图像，在可视区域内的所有物体都需要显示，而且每一物体都需要看上去有连续的二维投影，但对于触觉图像这部分，通常情况下只需要单独显示接触区域内的力就足够了。

润色触觉图像的软件接受物理模型的输出，产生驱动接口设备的命令。在“MIT 砂纸”这个二维带有力显示的游戏杆实例中，通过使用一个简单的规则成功地实现了触摸粗糙表面感觉的传递。即使采用如此简单算法产生的触觉图像，如果用户还有视觉或听觉的一致输入，则对于用户来说，这样的 VR 交互也是足够逼真的，因此，触觉图像的生成算法很大程度上依赖于特定的应用以及显示设备的能力上。

由于 VR 中触觉/力反馈通道处于研究的早期阶段，几乎没有专门为产生触觉可视图像而设计的软件。

3. 结论

总之，鼠标和空间球是最简单的触觉接口(参见图 12.17)，并被广泛地用于与计算机的交互上。不带力反应的位置感知手套与外衣在市场上已有产品，但主要是用于研究目的。在力反应设备中，基于地面的装置(如游戏杆)已被使用。带力反应的外廓设备相对要难设计一些，只有少量用于研究目的。触觉显示则更是具有设计难度的挑战，因为它需要皮肤上高密度的感受器以感受刺激。现在已有一些对手指的触觉刺激器，包括形状改变、电神经刺激及振动阵列，但还没有一种可以产生令人感觉逼真的触觉图像，它们都无法实际应用。



图 12.17 典型的空间球

目前，触觉接口只能产生与真实环境非常粗糙近似的交互刺激，但这并不意味着通过触觉接口产生的人工触感对使用者来说就必定不真实。我们可以以看电视或玩电子游戏的人工视觉感受为例，在现实世界中，视觉刺激在空间和时间上是连续的，但对于这些视觉接口来说，则是以 30 帧/秒的速度投影图像。我们之所以有栩栩如生的真实感就是因为我们利用了人体视觉器官的能力极限。因此，同样地我们可以利用人体触觉系统的能力极限，对某些特定的任务进行必要

的简化。总之，产生人工触感时，人体触觉系统的生物机械性能、运动传感性能及认知能力等仍需要进行大量的人体学研究。

12.7 虚拟场景的生成

通过前面几节的讨论，我们对虚拟现实的接口技术有了基本的了解。但是，之所以称为虚拟现实，首先必须由计算机产生与“现实”逼真的场景，即虚拟环境，然后才有如何实现与虚拟环境“交互”，如何达到“沉浸”，以及实现“联想”等。虚拟场景的生成主要包括两方面内容：虚拟环境的建模及三维图形显示。首先讨论如何建立场景的静态模型，重点说明与一般的计算机三维绘图的区别；然后讨论场景动态模型的建立，因为在虚拟现实环境中，不但存在人与计算机之间的交互，而且环境所包括的实体之间也可能发生交互作用。

12.7.1 虚拟场景静态模型的建立

尽管现实环境中可能有的物体千差万别，但在虚拟现实环境中均将其作为“对象”，需要显示的以“几何对象”加以描述，这是场景的基本元素，其形状特征在计算机内部一般都是用多边形来表示，这也是计算机图形学的传统方法。几何对象的完整描述应包括两方面，其一称为“静态模型”，它描述对象的静态特征，包括几何尺寸、形状特征、位置、方向、材料及有关属性；其二称为“动态模型”，它描述对象的动态属性，包括其运动、行为及其约束条件等。本节讨论静态模型的建立。

静态模型的描述一般分为：几何模型、光照模型、灰度模型、纹理处理、渲染效果等。显然，所有这一切如果要依靠用户自己用编程语言来实现，建立一个哪怕是非常简单的虚拟现实环境也是一件工作量巨大的事情。幸而由于世界各国研究人员的长期努力，我们可以采用商用软件来实现，甚至可以在商用模型库的基础上进行适当的开发。

这里介绍广泛得到使用的 OpenGL。它来源于 SGI 公司在 1982 年开发的 IRISGL，该图库的 3D 应用有 1500 个以上。OpenGL 继承了这些成果，保留了 IRISGL 全部绘图功能，新的应用则直接用 OpenGL 编写，其应用编程接口（application programming interface，API）对用户来说是中性的，也就是说，它支持多种工业标准，能在几乎所有现时的计算机系统上运行，其源程序可以不加修改地在不同的计算机上编译执行，包括 3-Dlabs，AccelGraghic，AT&T，Cirrus Logic，Cray Research，Daikin，Digital Equipment，Evans&Sutherland，Harris Computer，Hitachi，IBM，Intel，Intergragh，Japan Radio Company，MediaVision，Microsoft，NEC，NeTpowerm，Samsung，Sony 以及 Silicon Graghics 等。这些公司联手支持将 OpenGL 开发成为开放的国际图形标准，因此 OpenGL 在 2D 和 3D 图形应用开发中得到广泛的应用。

OpenGL 提供的图形功能范围十分广泛，从简单的几何点、线，多边形填充的绘图，到 NURBUS 曲面的着光及纹理映射等子例程，多达 250 个，可供用户开发调用。

例如，如果用户想构造一个具有光照、阴影且光顺的多面体，只需要交互式地选择“几何造型（geometric modeling）”，在菜单的驱动下构造出所需要的几何多面体的各个平面；然

后，选择“着光状态(lighting state)”设置材料特性的值；再打开“光滑阴影(smooth shading)”设置视点变换；对每个多边形，定义其当前的表面法向量以确定其姿态。这样，就可以根据多边形的方位及其相互之间的关系来确定多面体的顶点。OpenGL 状态机对其加以处理，并将这些多边形写进主缓冲器(frame buffer)。进一步，可打开深度缓冲器(depth buffer)，消除隐藏线和隐藏面，进行纹理映射和渲染(如加上背景颜色一片云彩、雾等)。所有这一切均可在交互式环境中实现。最后，一旦完成，就可以将其保存起来，需要时可再次调用。OpenGL 的体系结构可用图 12.18 表示。

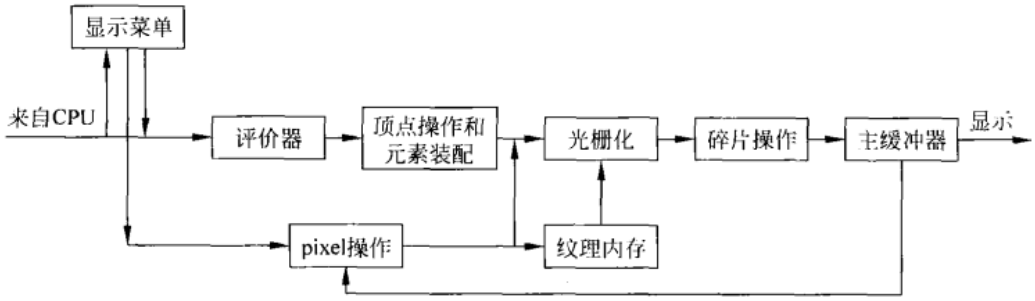


图 12.18 OpenGL 结构图

用户与 OpenGL 交互实际上是通过客户/服务器模式进行的。客户应用发出命令，OpenGL 服务器接收命令，并加以解释和处理。客户与服务器可以位于同一台计算机上，也可以是不同的计算机，因此，OpenGL 是网络透明的(参见图 12.19)。

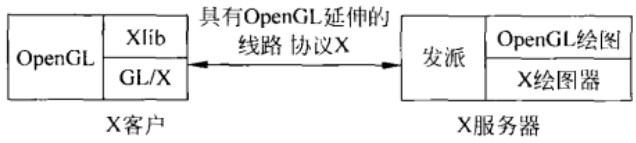


图 12.19 OpenGL 的工作模式

如前所述，OpenGL 独立于计算机硬件、视窗系统和操作系统，因而可在许多类型的计算机平台上方便地移植(参见图 12.20)，而且在实际应用时，可从通用的编程语言如 C,C++,ADA,FORTRAN 直接进行调用。

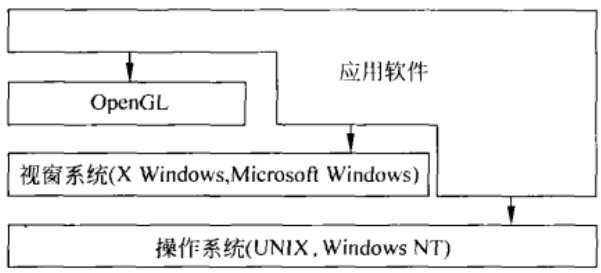


图 12.20 OpenGL 的可扩充性

在实际应用时，目前有数量相当多的应用软件开发工具包可供选择，而不必用通用编程语言来直接调用 OpenGL，比较典型的有：WTK(world tool kit)，MR (minimal reality tool kit)，Open Inventor 以及 VRML(virtual reality modeling language)等。特别是 Open

Inventor，它也是 SGI 公司开发的，因而与 OpenGL 具有紧密集成关系，并被选为基于万维网的虚拟现实建模语言 VRML 的基础。

当然，采用多边形近似的办法表示物体时，要求多边形要足够小，才能生成达到较好的效果。然而，由于 VR 系统要求图形刷新的速度最小不低于 15 幅/秒，计算机每秒钟处理的多边形数量有限，因此，不能仅仅依靠增加多边形的数量来提高场景效果。例如，一些形状非常复杂的如树木、花草、雕塑等，如果要用三维模型表示，则需要大量的多边形才能表示出来。实际上，这些背景物体并不一定要非常精确地表示。在 VR 实现的研究中，人们采用事先用扫描仪将这些复杂的对象图像输入到计算机中，将其贴在一个平面上并放置在场景中，实时动态显示时，令该平面的法向量始终指向观察点。这样，虽然造成这些背景物体无论从哪个角度看都只有一种形态，但由于它们表示的是背景，而背景在动态刷新过程中的这种效果并不会给视觉效果产生显著影响，但大大加快了图形刷新的速度，因而在目前的 VR 系统中得到广泛的应用。

12.7.2 虚拟场景动态模型的建立

虚拟场景的动态模型建立主要涉及两方面的内容，其一是如何确定场景中每一个对象运动过程中的坐标(包括绝对坐标与相对坐标)，其二是当视点变化时场景如何动态刷新。

实际上，在场景创建时，对每一个对象不仅要定义其绝对坐标系，也称为世界坐标系(the world coordinate)，而且要定义其本身的对象坐标系，也称为局部坐标系(the local coordinate)。当对象运动时，局部坐标系的位置根据运动方程在世界坐标系中发生相应的变化，但描述对象的图形元素如点、线、面等虽然随坐标系移动，但方向、姿态均不变化；另一方面，对象移动时，它与场景中的其他对象之间的相对位置、方向、姿态也要发生相应的变化。为此，Denavit，Hartenberg 和 Fu，Foley 等人研究解决了三维空间中对象平移、旋转和缩放问题。基本方法是采用如下 4×4 的齐次坐标变换阵进行相应的坐标变换，即

$$\boldsymbol{T}_{3d}=\begin{bmatrix}A_{11} & A_{12} & A_{13} & B_{14}\\ A_{21} & A_{22} & A_{23} & B_{24}\\ A_{31} & A_{32} & A_{33} & B_{34}\\ A_{41} & A_{42} & A_{43} & B_{44}\end{bmatrix}=\begin{bmatrix}\boldsymbol{A}_{3\times3} & \boldsymbol{B}_{3\times1}\\ \boldsymbol{A}_{1\times3} & \boldsymbol{B}_{1\times1}\end{bmatrix} \tag{12.15}$$

其中，$\boldsymbol{A}_{3\times3}$ 产生比例、旋转、错切等几何变换，$\boldsymbol{A}_{1\times3}$ 产生平移变换，$\boldsymbol{B}_{3\times1}$ 为投影变换，而 $\boldsymbol{B}_{1\times1}=B_{44}$ 产生整体比例变换。例如，若将对象 M 转换到对象 N 的坐标系，其转换矩阵可用 $\boldsymbol{T}_{N\leftarrow M}$ 来表示为

$$\boldsymbol{T}_{N\leftarrow M}=\begin{bmatrix}\boldsymbol{A}_{3\times3} & \boldsymbol{B}_{3\times1}\\ \boldsymbol{0}_{3\times1} & 1\end{bmatrix}$$

该变换阵不仅将对象的平移与旋转等变换统一起来，而且若要进行逆变换也易于处理，其逆变换阵为

$$\boldsymbol{T}_{M\leftarrow N}=\begin{bmatrix}\boldsymbol{A}_{3\times3}^{\mathrm{T}} & -\boldsymbol{A}_{3\times3}^{\mathrm{T}}\boldsymbol{B}_{3\times1}\\ \boldsymbol{0}_{1\times3} & 1\end{bmatrix} \tag{12.16}$$

由于实际环境的复杂性，一般将环境对象定义成递阶结构，每一层的对象称为节点(node)。最上层的节点称为“父”节点，下一层称为“子”节点，依次下去，相邻的两层构成“父

子”关系，从而形成节点树。对每个节点不但定义其绝对坐标系，而且也定义其局部坐标系，如果父节点的坐标系发生了变化，则子节点的空间坐标就易于计算出来，相反，子节点的空间位置发生变化，如果其父节点不受其约束，它仍可独立进行坐标变换。

动态模型的另一方面表现为视点变化所引起的视景变化。这相当于将对象的三维空间坐标映射到位于视点的摄像机的二维屏幕上。在 VR 中，一般采用“透视投影”方法，其基本原理见图 12.21。

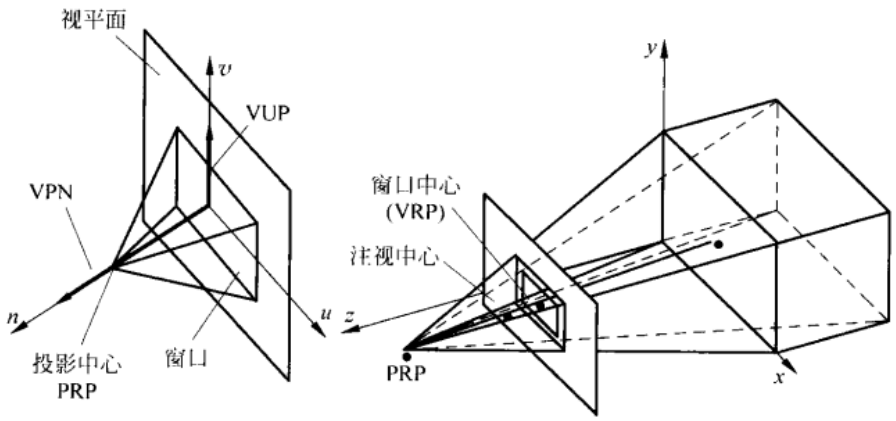


图 12.21 虚拟摄像机透视投影原理

虚拟摄像机位于投影中心(PRP)，由 PRP 到三维物体各点的连线与视平面的交点就形成了物体的透视投影。显然，离 PRP 近，投影就大，反之，投影就小。由于 VR 系统的视场角有限，需要在视平面定义一个二维窗口，其高为 H，宽为 W，从而确定可视区域。为了建立与对象坐标系的联系，需要定义“观察坐标系”(u，v，n)，坐标原点位于窗口中心，也称为“观察参考点”(VRP)；视平面的法向量称为“观察平面法向量”(VPN)，确定窗口上下位置的向量称为“向上向量”(VUP)。虚拟摄像机摄像时，投影平面应垂直于世界坐标系的 Z 轴。这样，当虚拟摄像机在三维空间移动时，只要给定 PRP、VUP、VRP、VPN 及相关参数，即可将世界坐标系中的位置转换到摄像机的观察坐标系中的位置，从而实现投影变换。描述虚拟摄像机的空间位置的上述参数由位置跟踪通道提供。当然这里仅就透视投影技术的原理进行简单说明，对不同的 VR 系统其具体实现的方法可能有很大的不同。

12.8 典型的虚拟现实系统软件

软件技术是虚拟现实系统的“灵魂”。它们的主要功能是高效地产生与真实物体形似、质似和神似的虚拟物体和虚拟环境，从而与硬件设备相得益彰地创造出“身临其境”的效果。“形似”就是建立与真实物体在外观上相似的虚拟对象，即“几何建模”；“质似”就是与真实物体在质地上的相似，即“物理建模”；“神似”是指与真实物体在特征、气质上的相似。

现在，市场上比较多见的是建模软件，它们是一种用作三维造型和几何建模的普及型软件，除了建模外，在表面物理属性的处理功能上也不断加强，如 3DS MAX、MAYA、AutoCAD 等，在广告设计、动画制作等领域已经大显身手。而实时仿真软件则是虚拟现实

技术的专用产品，它们一般用作整个虚拟环境的集成和对虚拟对象模型进行驱动，产生各种动态的效果。由它们制作的虚拟环境与普通的三维动画不同之处在于后者只是一部内容和播放顺序完全固定的动画片，制作完成后，用户也就不能再做变化和更动；而由仿真软件制作的虚拟场景却是一种交互的环境，能根据用户的输入要求，随意变换视点和播放的内容及顺序，给用户强烈的亲身参与感和沉浸感。

1. WorldToolKit 软件

美国 EAI/Sense8 公司的 WorldToolKit 是较为成熟的工具包。WorldToolKit 是一种简洁的跨平台软件开发系统，可用于科学和商业领域建立高性能的、实时的、综合 3D 工程。WorldToolKit 具有很强的功能及终端用户工具，它可用来建立和管理一个项目并使之商业化。一个高水平的应用程序界面(API)应该能让用户按需要快捷地建模、开发及重新构造应用程序。WorldToolKit 也支持基于网络的分布式模拟环境以及工业上大量的界面设备，如头盔显示器、跟踪器和导航控制器。

WorldToolKit 主要特点如下：

1) 为性能而设计

WorldToolKit 的算法设计使画面高品质得到根本的保障。这种高效的视觉数字显示提高了运行、控制和适应能力，它的特点是高效传输数据及细节分辨。

2) 为开发而强化

WorldToolKit 提供了强大的功能，它可以开发出最复杂的应用程序，还能提高一个组织的生产效率。一个用 C 代码编写的对象定位库提供给用户 1 000 多个高级语言的函数，可用来构造、交互和控制实时模拟系统。一个函数调用相当于执行 1 000 行代码，这将缩短产品开发时间。

WorldToolKit 被规划为包括 the universe 在内的 20 多个类，它们分别管理模拟系统、几何对象、视点、传感器、路径、光源和其他项目。附加函数用于器件实例化、显示设置、碰撞检测、从文件装入几何对象、动态几何构造、定义对象动作和控制绘制。

3) 开放性和系统优化

WorldToolKit 使 OpenVRTM 的理论成为现实，它提供了一个工具可简捷地跨过不同的平台，包括 SGI、Evans 和 Sutherland，Sun，HP 和 Intel。优化的功能使它可以对付每一个平台界面，它直接通过连续的系统图片库使最快速的传输图片成为可能。另外，WorldToolKit 支持多种输入输出设备，并且允许用户修改 C 代码——如设备驱动器、文件阅读器和绘图例行程序，也允许它和多种信息源进行交互。

4) 高级函数调用

WorldToolKit 包含的函数可用来实例化和询问知名的设备，如 Polhemus FASTRAK 和 Ascension Bird。

2. Vega 软件

Vega 是 MultiGen-Paradigm 公司开发实时视觉和听觉仿真、虚拟现实应用的软件环境。它把先进的仿真功能和易用的工具结合到一起，创建了一种使用简单且具创造力的体系结构，来创建、编辑和运行高性能的实时应用，用于实时视觉模拟，虚拟现实和普通视觉应

用。Vega 将先进的模拟功能和易用工具相结合，对于复杂的应用，能够提供快速、方便的建立、编辑和驱动工具。Vega 能显著地提高工作效率，同时大幅度减少源代码开发时间。

1）AudioWorks2 声音模拟模块

AudioWorks2 声音模拟模块针对多个对象、多个听点和模拟对象的物理特性，连续实时地处理声音波形，在开阔地带提供空间感极强的三维声音。它提供了一个基于物理特性，包括距离衰减、多普勒漂移和传输延迟的无回声声音生成模型。

2）Marine 海洋模块

Vega 海洋模块为逼真海洋仿真提供了所必须具有的海洋特殊效果，使得在 Windows NT 和 UNIX 操作系统平台上开发海洋仿真应用。所有的海洋效果都可以在 LynX 图形界面中设置，或通过 C 应用程序接口加以控制。

3）Symbology 动态仪表和动态字符模块

Vega 的动态仪表和动态字符模块可以满足虚拟现实应用和实时工程仿真中仪表和图形状态显示的需要。已经建好模型的图形对象可通过 LynX 直接装载到 Vega 应用，利用建好的模型来生成相应的仪表和字符显示。

4）NSL 导航及信号灯模块

导航及信号灯模块使用户可以使用光点逼真地生成导航和信号灯系统。在实时运行环境里，可以创建、配置光点，并控制其方向性、直径、范围和能见范围等物理特性。

5）Light Lobes 光束模块

Vega 光束模块使用独特的分析技术，在对硬件光源和投射纹理没有负面影响的情况下，可以任意创建和生成逼真的可移动的场景照明。NLDC-Vega 非线性失真矫正模块，为用户提供了对 Vega 应用进行静态或动态失真矫正的功能。通过 LynX 图形界面，用户可以创建投影表面的软件描述，并建立起在球幕内的投影器位置和眼点之间的关系。VCR-Vega 的记录及重放模块为记录和重放显示的场景提供了一种点击式环境，包括各种 Vega 类，如对象、观察者、特殊效果以及用户自定义的类。

6）CloudScape 云景模块

CloudScape VR 模块可以实时生成量化的、含辐射度的三维云，并支持云现象的其他模块，与数据库一起可以生成各种气候条件下的云、武器效果、炸弹生成的灰尘等其他的战场环境。

7）DI-Guy 人体模拟模块

Vega 的人体模拟模块为仿真环境中添加了好像有生命的动画人物。每个人物可以根据简单的命令，逼真地在场景中移动。即使从一种活动转成另一种活动时，其转换过程也是平滑和自然的。

8）SSVO 载体对象管理模块

采用载体对象模块，大大简化了将多个对象组成一个更高一层对象的工作量，使得建立复杂系统铰链关系变得更加容易，并可以交互地将各种部件附属于载体对象，对建立的铰链关系进行预览。

9）Immersive 浸入式环境模块

Vega 浸入式环境模块支持多个观察者在完全浸入的环境中，在 CAVE 系统中任意设置投影显示面数。通过 LynX 图形界面进行系统和硬件平台的设置，并支持大量的立体显

示设备。

10) Sensors 传感器模拟模块

通过传感器模拟模块,可以实时模拟基于其物理特性的从远红外到可见光的各种传感器,各种雷达成像模式的雷达显示画面。

11) Special Effects 特殊效果模块

Vega 特殊效果模块通过使用各种不同的实时技术,从基于没有纹理的硬件的加阴影几何体到借助纹理分页技术的复杂的粒子动画,来产生实时应用中的三维特殊效果。

12) DIS/HLA 分布交互式仿真模块

通过扩展 LynX 来提供 DIS/HLA 操作,分布交互式仿真模块在不需要任何编程的情况下,加速了分布交互式仿真应用的开发过程。VR-Link 提供分布交互式仿真网络接口软件,它是 Vega 分布交互式仿真网络通信的基础。

13) LADBM 大规模数据库管理模块

采用大规模数据库管理模块,使得管理大型复杂的数据库变得非常容易。大规模数据库管理模块使用双精度内核,可动态地重新设定移动感兴趣区域的地面坐标系原点,使观察者始终处于要显示的数据库附近。

3. MultiGen Creator

MultiGen Creator 是 MultiGen-Paradigm 公司的软件工具集,用于产生高优化、高精度的实时 3D 内容,用于视景仿真、交互式游戏、城市仿真以及其他领域。

该工具集提供交互式的实时 3D 建模能力。MultiGen Creator 软件包是交互式的、高度自动化的软件,它可以高效、实时地产生 3D 数据库而不会影响可视质量。它在满足实时性的前提下生成面向仿真的、逼真性好的大面积场景。它可为二十五种不同类型的图像发生器提供建模系统及工具,它的 OpenFlight 格式在实时三维领域成为最流行的图像格式,并成为仿真领域的行业标准,由以下几部分组成:

1) Creator Pro

MultiGen Creator Pro 是一套实时三维建模工具,能够满足视景仿真、交互式游戏开发、城市仿真以及其他的应用领域。Creator Pro 是唯一将多边形建模、矢量建模和地形生成集成在一个软件包中的手动建模工具,它能进行矢量编辑和建模、地形表面生成。

2) Terrain Pro

Terrain Pro 是一种快速创建大面积地形数据库的工具,它可以使地形精度接近真实世界,并带有高逼真度三维文化特征及图像特征。

3) Road Pro

Road Pro 扩展了 Terrain Pro 的功能,利用高级算法生成路面特征,以满足驾驶仿真的需要。

4) Interoperability Pro

Interoperability Pro 提供了用于读、写及生成标准格式数据的工具,主要用于 SAF 系统、雷达及红外传感器的仿真。

5) SmartScene

SmartScene 是将实时 3D 技术应用于训练,它使工作者完全融入虚拟环境过程成为

可能。

6) OpenFlight

OpenFlight 为 MultiGen 数据库的格式，它是一个分层的数据结构。OpenFlight 使用几何层次结构和属性来描述三维物体，它采用层次结构对物体进行描述，可以保证对物体顶点、面的控制。

12.9 小结

正如我们在 12.2 节中提到的，虚拟现实技术具有广阔的发展与应用前景，但目前尚处于“婴儿”阶段。尽管虚拟现实技术已经取得了长足的进步，但仍受到以下两个基本问题的制约。

(1) 关于人的感知和认知模型的研究

尽管我们每时每刻感知和认知周围的客观世界，但是，我们却并不清楚人类到底是如何感知和认知的。譬如，当我们的手指碰上 220V 的电线时，那种剧烈的反应到底是如何发生的？那种灼伤的感觉如何描述？又如，人对语音的认知能力是如何起作用的？采用何种技术才能使计算机具有人类的理解力？等等。当然，对人类自身的认识不仅仅是 VR 的任务，包括生理学、心理学、行为科学、认知科学等，然而，VR 不仅要认识它，还要用其他的技术手段去构造它，因此，无疑还有很长的路，甚至是一条没有尽头的路。

(2) 关于虚拟现实的“真实性”的评价准则的研究

这一问题与前一问题是相关的。现在世界上已有种类各异、价格各异、技术水平各异的“虚拟现实”产品被研制出来，我们可以从技术的角度对不同产品的性能进行评价，但是，我们要问，这些产品真正达到了“虚拟现实”吗？它与“真正现实(real reality)”有多大的差距？甚至到底如何评价它们之间差距的准则也难以确定。由于不同的人对“真实性(realism)”的感知很不相同，目前尚只能针对具体的 VR 系统由不同的人参加调试，以多数人认为“满意”为评价准则。

尽管如此，虚拟现实技术已经而且将继续迅速发展起来，不仅因为它可为人类提供栩栩如生的文化娱乐环境，更是因为它为研究和探索宏观世界、微观世界以及由于种种原因难以直接面对的真实对象提供了有效手段。

参考文献

[1] Ellis S R. What Are Virtual Environments?. Computer Graphics and applications, 1994, 1

[2] Nathaniel I, Durlach and Mavor S. Virtual Reality, Scientific and Technological Challenges. National Academy Press, Washington, D. C. 1995

[3] 汪成为，高文，王行仁. 灵镜(虚拟现实)技术的理论、实现及应用. 北京：清华大学出版社，南宁：广西科学技术出版社，1996

[4] Carr K, England R. Simulated and Virtual Realities (Elements of perception). Burgess Science Press, Basingstoke, Great Britain, 1995

[5] 曾芬芳. 虚拟现实技术. 上海：上海交通大学出版社，1997
[6] 吴启迪. 系统仿真与虚拟现实. 北京：化学工业出版社，2002
[7] 姜国华. 虚拟现实技术及在航空航天中的应用. 北京：国防工业出版社，2007
[8] Edgar G K, Pope J C D and Craig I. Visual accommodation problems with head-up and helmet-mounted displays?, Displays. 1994, 65: 68～75
[9] Ando S, Shinoda H, Ogawa K and Mitsuyama S. A three-dimensional sound localization sensor system based on the spatio-temporal gradient method. Transcripts of the Society of Instrument and Control Engineers, 1993, 29(5)
[10] Nielsen S H. Auditory distance perception in different rooms. Journal of the Audio Engineering Society, 1993, 41(10)
[11] So R H Y and Griffin M J. Effects of lags on human operator transfer functions with head-coupled systems. Aviation, Space and Environmental Medicine, 1995
[12] Janin A L, Mizell D W and Caudell T P. Calibration of head-mounted displays for augmented reality applications. In Proceedings of the IEEE VRAIS'93, Seattle, WA, New York, IEEE, 1993, 246～255
[13] Hirota K and Hirose M. Providing Force Feedback in Virtual Environment. Computer Graphics and Application, 1995, 15(5)
[14] Vacca J. VRML Bringing Virtual Reality to the Internet. Academic Press, Inc., 1996
[15] http://www.ycrs.net.cn/V8.htm

附录 1
实验指示书及仿真源程序

实验一　面向方程的数值积分方法仿真

一、实验目的

运用 CSS01 仿真程序解题，培养阅读及修改仿真程序的能力，学习并了解仿真程序的结构及特点。通过实验，加深理解四阶龙格-库塔法的原理及其稳定域。

二、实验内容

修改 CSS01.C 仿真程序，对如下系统进行仿真。

(1) 线性定常系统

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix}=\begin{bmatrix}-20 & 1 & 0\\ 0 & -200 & 1\\ -600 & 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}+\begin{bmatrix}0\\ 0\\ 600\end{bmatrix}u$$

$$\begin{bmatrix}x_1(0)\\ x_2(0)\\ x_3(0)\end{bmatrix}=\begin{bmatrix}0\\ 0\\ 0\end{bmatrix},\quad u=1(t)$$

(2) 非线性系统

$$\begin{cases}\dfrac{\mathrm{d}x(t)}{\mathrm{d}t}=rx(t)-ax(t)y(t)\\ \dfrac{\mathrm{d}y(t)}{\mathrm{d}t}=-sy(t)+bx(t)y(t)\end{cases}$$

其中

$$r=0.001,\qquad a=2\times10^{6},\qquad x(0)=12\,000$$
$$s=0.01,\qquad b=1\times10^{-6},\qquad y(0)=600$$

三、预习要求

(1) 根据实验内容，修改 CSS01.C 有关程序。

(2) 为保证仿真的稳定性，分析第 1 题，其最大仿真步长为多少？

(3) 写出实验步骤。

四、实验要求

改变仿真步长，观察上述两个系统步长多大时仿真变得不稳定？仿真步长多大为宜(即进一步减少步长，精度没有显著提高)？

五、实验报告

实验完成后，要写出实验报告，内容包括：

(1) 预习报告；

(2) 实验步骤及说明；

(3) 实验所用仿真程序清单；

(4) 实验结果及分析。

下面给出 CSS01.C 的源程序。CSS01 是面向微分方程的仿真程序，程序中用四阶龙格-库塔法进行数值积分运算。

六、CSS01.C 源程序清单

```
/* --------CSS01.C 源程序--------本程序用于四阶龙格-库塔法求解微分方程
   程序中变量说明
   y:   状态变量数组
   m:   保存状态变量最大值的数组
   e:   保存状态变量最小值的数组
   g:   状态变量导数值数组
   p:   未知参数数组
   z:   存放四阶龙格-库塔法计算的中间结果数组
   k:   四阶龙格-库塔系数数组
   w:   保存输出结果的数组
   a:   代替输出状态变量的数组
   t:   仿真时间
   t1:  仿真总时间
   t2:  积分步长
   t3:  输出点时间
   t4:  输出点间隔时间
   t7:  积分步长的一半
   l3:  表示首次运行的标志
   j8:  运行次数
   j9:  完成运行的次数
   n1:  被仿真系统的阶次
   n2:  未知参数的个数
   n3:  输出点数
   n4:  输出数组索引
   n5:  输出数组索引
------------------------------------------------------------------------------*/
#include<math.h>
#include<stdio.h>
#include<stdlib.h>
```

```
void inisub(void);
void runsub(void);
void intsub(void);
void stosub(void);
void outsub(void);
void modsub(void);
void repsub(void);
float t,t1,t2,t3,t4,t7;
int l3,j8,j9,n1,n2,n3,n4,n5;
float y[31],m[31],e[31],g[31],p[21],z[21],k[21][5],w[102][6],a[5];

main()
{
  int i;
  inisub();    /*变量初始化*/
  do
  {          /*输入控制参数 */
    if(l3! =1)
    {
        if(j8==1)  exit(0);
        printf("Input total simulation time T1,step T2=");
        do
        {
            scanf("%f,%f",&t1,&t2);
            if(t1<t2||t1<0||t2<0)
                printf("wrong number entered. re-input T1,T2=");
        }while(t1<t2||t1<0||t2<0);
        t7=t2/2.0;
        printf("Input the orders of system N1=");
        do
        {
            scanf("%d",&n1);
            if(n1<0||n1>20)
                printf("wrong number entered.. re-input N1=");
        }while(n1<0||n1>20);
        printf("Input the number of unknown parameters N2=");
        do
        {
          scanf("%d",&n2);
          if(n2<0||n2>20)
            printf("wrong number entered. re-input N2=");
        }while(n2<0||n2>20);
        printf("Input the number of output points N3=");
        do
        {
            scanf("%d",&n3);
            if(n3<2||n3>100)
                printf("wrong number entered. re-input N3=");
        }while(n3<2||n3>100);
        printf("Input the number of runs J8=");
        do
```

```
        {
            scanf("%d",&j8);
            if(j8<0)
                printf("wrong number entered. re-input J1=");
        }while(j8<0);
        if(j8>=2)
        {
            l3=1;                          /*设置再一次运行的标志*/
            j9=j9+1;
            if(j9>j8) exit(0);             /*已经完成的运行次数*/
        }
      }
      else
      {
          j9=j9+1;
          if(j9>j8) exit(0);
      }
      printf("Input initial values of state variables =");
      for(i=1; i<=n1; i++)
          scanf("%f",&y[i]);
      printf("Input the values of parameters=");
      for(i=1; i<=n2; i++)
          scanf("%f",&p[i]);
      printf("\n");
      runsub();
      outsub();
    }while(1);
}
void runsub(void)
{
  int i;
  for(i=1; i<=n1; i++)
    z[i]=y[i];                             /*保存状态变量*/
  n4=1;
  t=0.0;
  t4=t1/(n3-1);
  if(t2>=t4)  t2=t4;
  t3=t4;
  t7=0.5*t2;
  modsub();                                /*计算微分方程*/
  stosub();                                /*存储结果*/
  do
  {
    do intsub(); while(t<t3);              /*完成一个计算点*/
    n4++;
    stosub();
    t3=n4*t4;
  }while(t<t1);                            /*完成所有计算点*/
  for(i=1; i<=n1; i++)
      y[i]=z[i];                           /*恢复状态变量*/
}
```

```
void inisub(void)                          /*本函数用于程序对变量初始化(为0)*/
{
    int i,j;
    for(i=0; i<=20; i++)
    {
      p[i]=0.0;
      g[i]=0.0;
      z[i]=0.0;
      for(j=0; j<=4; j++)
      {
          k[i][j]=0.0;
          a[j]=0.0;
      }
    }
    t=0.0;
    l3=0.0;
    for(i=0; i<=30; i++)
    {
      y[i]=0.0;
      m[i]=0.0;
      e[i]=0.0;
    }
    j9=0;
    for(i=1; i<=101; i++)
    {
      for(j=0; j<=5; j++)
          w[i][j]=0.0;
    }
}
void intsub(void)                          /*本函数用于龙格-库塔积分计算*/
{
    int i;
    for(i=1; i<=n1; i++)
    {
        k[i][2]=g[i]*t7;
        k[i][1]=y[i];
        y[i]=k[i][1]+k[i][2];
    }
    t=t+t7;
    modsub();
    for(i=1; i<=n1; i++)
    {
        k[i][3]=g[i]*t7;
        y[i]=k[i][1]+k[i][3];
    }
    modsub();
    for(i=1; i<=n1; i++)
    {
        k[i][4]=g[i]*t2;
        y[i]=k[i][1]+k[i][4];
    }
    t=t+t7;
    modsub();
    for(i=1; i<=n1; i++)
```

```
        y[i]=k[i][1]+(2*k[i][2]+2*k[i][4]+4*k[i][3]+g[i]*t2)/6;
    modsub();
}
void stosub(void)                                   /*本函数用于记录仿真结果*/
{
    int i;                                          /*每个输出点的时间和状态变量值 */
    n5=n4-1;
    w[n5][1]=t;
    repsub();                                       /*获得所需要的输出状态变量的值*/
    for(i=1; i<=30; i++)
    {
        if(y[i]>m[i]) m[i]=y[i];
        if(y[i]<e[i]) e[i]=y[i];
    }
    for(i=2; i<=5; i++)
        w[n5][i]=a[i-1];
}
/*------------------输出函数-------------------------------------------------------------*/
void outsub(void)
{
    int i,j;
    printf("Y[I]                Y[I]max                Y[I]min\n");
    for(i=1; i<=30; i++)
        if(m[i]! =0||e[i]! =0)
            printf("%2d              %15.5f        %15.5f\n",i,m[i],e[i]);
    printf("SIMULATION STEP T2=%15.5f\n",t2);
    printf("          T            A[1]          A[2]          A[3]          A[4]\n");
    for(i=0; i<=n5; i++)
    {
        for(j=1; j<=5; j++)
            printf("%13.5f",w[i][j]);
        printf("\n");
    }
}
/*-----------------------------定义被仿真的系统模型-------------------------------------*/
/*          若是新系统,表示系统的方程应在此定义,以便运行前编译                        */
void modsub(void)
{
    y[21]=y[1]*cos(y[2]);
    y[22]=y[1]*sin(y[2]);
    g[1]=y[3];
    g[2]=y[4];
    g[3]=-401408/(y[1]*y[1])+y[1]*y[4]*y[4];
    g[4]=-2*y[3]*y[4]/y[1];
}
/*-------------------------选择输出状态变量的函数-----------------------------------*/
void repsub(void)
{
    a[1]=y[21];
    a[2]=y[22];
    a[3]=y[1];
    a[4]=y[2];
}
```

实验二 面向结构图的离散相似法仿真

一、实验目的

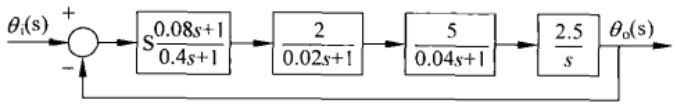
运用 CSS02 仿真程序解题。掌握时域离散相似法的仿真模型实现；学习含有非线性环节的仿真方法；加深理解离散相似法的原理及特点。

二、实验内容

利用 CSS02.C 仿真程序，对如下小功率随动系统进行仿真研究。

(1) 线性系统(如附图 1 所示)

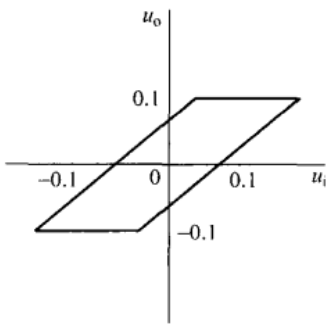
仿真 $\theta_i=10\cdot 1(t)$作用下，输出 θ_0 的跟踪过程。



附图 1 小功率随动系统

(2) 非线性系统

考虑如上线性系统中，第 4 个环节后有一回环非线性环节，仿真在 $\theta_i=10\cdot 1(t)$作用下，θ_0 的跟踪过程。非线性环节的参数如附图 2 所示。



附图 2 回环非线性环节特性

三、预习要求

(1) 计算线性系统开环切割频率 ω_0 及其过渡过程时间的估计值(5%)，并按离散相似法选择仿真步长的原则预选仿真步长。

(2) 考虑回环特性时，计算系统的自激振荡频率及振荡幅值。

(3) 列出仿真步骤及说明。

四、实验报告

根据仿真计算结果分析系统的自激振荡频率及振荡幅值，并将其与理论分析的结果加以比较。比较数值积分法与离散相似法的优缺点。

下面给出 CSS02 源程序。CSS02 是非线性系统仿真程序。包含常用的典型非线性，面向结构图输入系统。程序采用离散相似法进行数值积分运算。

五、CSS02.C 源程序清单

```
/*****************************************************************/
/*              CSS02.C 源程序 */
/* 说明：z[i]表示非线性环节 */
/*          如果  z[i]是奇数，则在编号为 i 的环节前有非线性 */
/*          如果  z[i]是偶数，则在编号为 i 的环节后有非线性 */
/*          如果  z[i]等于 0，则在编号为 i 的环节前或后均无非线性 */
/*****************************************************************/
#include "stdio.h"
```

```
#include "math.h"
#define lpn stdout
/*------------------------ 变量定义 ------------------------*/
typedef float REAL[21];
typedef int   INT[21];
REAL a,b,c,d,e,f,g,x,y;
REAL w,k,s,o,q,u,v;
REAL yout[100];
float tl[100],p[21][22];
float t,step,rm;
INT h,z;
int n0,n1,n2,n3,n4,n;
int l1,l2;
/*------------------------ 函数定义------------------------*/
void subinput_s();
void subinput_b();
void subcoe();
void subhead();
void subdeu();
void subaux();
void subcrt();
void subout();
void sub1(int m , float c1);
void sub2(int m , float c1);
void sub3(int m , float c1);
/*------------------------ 主函数 ------------------------*/
main()
{
  int iq1,iq2;
  int i,j;
  subinput_s();
  subinput_b();
  subcoe();
  t=0.0;
  subhead();
tl[0]=0.0;
for(i=1; i<=n; i++)
      yout[0][i]=y[i];
y[0]=rm;
for(iq1=1; iq1<=l2; iq1++)
{
  for(iq2=1; iq2<=l1; iq2++)
  {
       for(i=1; i<=n; i++)
       {
            u[i]=0.0;
            for(j=0; j<=n; j++)
            u[i]+=p[i][j]*y[j];
       }
       subdeu();
       for(i=1; i<=n; i++)
```

```
        {
          v[i]=(u[i]-w[i])/step;
          w[i]=u[i];
          x[i]=e[i]*x[i]+f[i]*u[i]+g[i]*v[i];
          switch(h[i])
          {
            case 0:
            case 2:
            case 4:
              y[i]=x[i];
              break;
            case 1:
              y[i]=x[i]+b[i]*k[i]*u[i]+b[i]*k[i]*v[i]*step;
              break;
            case 3:
              y[i]=(b[i]-a[i])*x[i]+k[i]*u[i]+k[i]*v[i]*step;
              break;
            default:
              printf("** h[%d]=%d incorrect! \n",i,h[i]);
          }
        }
        t+=step;
        subaux();
      }
      subcrt();
      for(i=1; i<=n; i++)
          yout[iq1][i]=y[i];
      t1[iq1]=t;
      }
      subout();
      getch();
}
/*------------------------------------ 输出函数 suboutput(s)------------------------------------*/
void suboutput_s()
{
  printf("The order of system N=");
  scanf("%d",&n);
  printf("Input N0,N2,N3,N4=");
  scanf("%d,%d,%d,%d",&n0,&n2,&n3,&n4);
  printf("Enter the input R=");
  scanf("%f",&rm);
  printf("Enter the STEP=");
  scanf("%f",&step);
  printf("Enter L1,L2=");
  scanf("%d,%d",&l1,&l2);
}
/*------------------------------------ 输入函数 subinput(b)------------------------------------*/
void subinput_b()
{
  int i,j;
  int i1,j1;
```

```
    float p1;
    printf("Enter:\n        A, B, C, D, X, Y, Z, S \n");
    for(i=1; i<=n; i++)
    {
      printf("No. %d: ",i);
      scanf("%f,%f,%f,%f,%f,%f,%d,%f",&a[i],&b[i],&c[i],&d[i],&x[i],&y[i], &z[i],
&s[i]);
      for(j=0; j<=n; j++)
              p[i][j]=0.0;
    }
    printf("Enter link matrix:\nI, J , P   \n");
    while(1)
    {
      scanf("%d,%d,%f",&i1,&j1,&p1);
      if((i1>0&&i1<=n)&&(j1>=0&&j1<=n))       p[i1][j1]=p1;
      else  if(i1>n||j1>n)  printf("p[%d][%d] : i or j over range! retry.\n",i1,j1);
          else  break;
    }
}
/*---------------------------- 计算状态转移系数函数 subcoe----------------------------*/
void subcoe()
{
    int i;
    for(i=1; i<=n; i++)
    {
        if(a[i]==0.0)
        {
          if(d[i]==0.0)
          {
              h[i]=0;
              k[i]=c[i]/b[i];
              a[i]=0.0;
              b[i]=0.0;
          }
        }
        else
        {
            h[i]=1;
            k[i]=c[i]/b[i];
            a[i]=0.0;
            b[i]=d[i]/c[i];
        }
      }
      else
      {
          if(d[i]==0.0)
          {
              if(b[i]==0.0)
              {
                h[i]=4;
                k[i]=c[i]/a[i];
```

```
                a[i]=0.0;
                b[i]=0.0;
              }
              else
              {
                h[i]=2;
                k[i]=c[i]/b[i];
                a[i]=a[i]/b[i];
                b[i]=0.0;
              }
            }
            else
            {
              h[i]=3;
              k[i]=d[i]/b[i];
              a[i]=a[i]/b[i];
              b[i]=c[i]/d[i];
            }
        }
        w[i]=0.0;
        if(z[i]==6)     q[i]=y[i];
        else            q[i]=0.0;
        o[i]=0.0;
        switch(h[i])
        {
          case 0:
          case 1:
                  e[i]=1.0;
                  f[i]=k[i]*step;
                  g[i]=k[i]*step*step/2.0;
                  break;
          case 2:
          case 3:
                  e[i]=exp(-a[i]*step);
                  f[i]=k[i]*(1.0-e[i])/a[i];
                  g[i]=k[i]*(e[i]-1.0)/(a[i]*a[i])+k[i]*step/a[i];
                  break;
          case 4:
                  e[i]=0.0;
                  f[i]=k[i];
                  g[i]=0.0;
          }
        }
}
/*---------------------------------- 打印输出头函数 subhead----------------------------------*/
void subhead()
{
  switch(n0)
  {
    case 1:
        printf("  T                    Y(%d)  \n",n2);
```

```
            printf("   %15.5f      %15.5f      \n",t,y[n2]);
            break;
        case 2:
            printf("   T                    Y(%d)              Y(%d)      \n",n2,n3);
            printf("   %15.5f      %15.5f      %15.5f      \n",t,y[n2],y[n3]);
            break;
        case 3:
            printf("   T                    Y(%d)              Y(%d)              Y(%d)   \n",
                n2,n3,n4);
            printf("   %15.5f      %15.5f      %15.5f      %15.5f      \n",t,y[n2],y[n3],y[n4]);
            break;
        default:
            printf(" cannot output more than 3 varibles. \n");
        }
}
/*-------------------------------- 打印输出函数 subcrt--------------------------------------*/
void subcrt()
{
  switch(n0)
    {
    case 1:
        printf("   %15.5f      %15.5f      \n",t,y[n2]);
        break;
    case 2:
        printf("   %15.5f      %15.5f      %15.5f      \n",t,y[n2],y[n3]);
        break;
    case 3:
        printf("   %15.5f      %15.5f      %15.5f      %15.5f      \n",t,y[n2],y[n3],y[n4]);
        break;
    default:
        printf(" cannot output more than 3 varibles. \n");
    }
}
/*--------------------------------计算非线性函数 subdeu--------------------------------*/
void subdeu()
{
  int i;

  for(i=1; i<=n; i++)
  {
      switch(z[i])
      {
        case 1: sub1(i,s[i]); break;
        case 3: sub2(i,s[i]); break;
        case 5: sub3(i,s[i]); break;
        case 0:
        case 2:
        case 4:
        case 6:break;
        default: printf("z[%d] value incorrect. \n",i);
      }
```

```
    }
}
/*---------- 确定是否有非线性函数 subaux----------*/
void subaux()
{
  int i;
  for(i=1; i<=n; i++)
  {
      u[i]=y[i];
      switch(z[i])
      {
          case 2: sub1(i,s[i]); y[i]=u[i]; break;
          case 4: sub2(i,s[i]); y[i]=u[i]; break;
          case 6: sub3(i,s[i]); y[i]=u[i]; break;
          case 0:
          case 1:
          case 3:
          case 5: break;
          default: printf("z[%d] value incorrect.\n",i);
      }
  }
}
/*---------- 典型非线性环节函数 sub1----------*/
void sub1(int m,float c1)
{
  if(fabs(u[m])>=c1)
      if(u[m]>=c1)
          u[m]=c1;
      else
          u[m]=-c1;
  else
          u[m]=u[m];
}
void sub2(int m,float c1)
{
  if(fabs(u[m])>=c1)
      if(u[m]>=c1)
          u[m]-=c1;
      else
          u[m]+=c1;
  else
      u[m]=0.0;
}
void sub3(int m,float c1)
{
  if(u[m]<=o[m])
      if(u[m]<o[m])
          if(q[m]>=(u[m]+c1))
          {
              o[m]=u[m];
              u[m]+=c1;
```

```
                q[m]=u[m];
            }
            else
            {
                o[m]=u[m];
                u[m]=q[m];
                q[m]=u[m];
            }
        else
        {
            o[m]=u[m];
            u[m]=q[m];
            q[m]=u[m];
        }
    else
    {
        o[m]=u[m];
        if(q[m]<=(u[m]-c1))
        {
            u[m]-=c1;
            q[m]=u[m];
        }
        else
        {
            u[m]=q[m];
            q[m]=u[m];
        }
    }
}
/* ----------------------------输出函数 subout ---------------------------------------- */
void subout()
{
    int i;
    switch(n0)
    {
            case 1:
              printf("T          Y(%d)\n",n2);
              break;
            case 2:
              printf("T          Y(%d)       Y(%d)     \n",n2,n3);
              break;
            case 3:
              printf("T          Y(%d)       Y(%d)     Y(%d)  \n" ,n2,n3,n4);
              break;
            default:
              printf(" cannot output more than 3 varibles.\n");
    }
    for(i=0; i<=l2; i++)
    {
        switch(n0)
        {
```

```
                case 1:
                  printf("%15.5f  %15.5f  \n",t1[i],yout[i][n2]);
                  break;
                case 2:
                  printf("%15.5f  %15.5f  %15.5f  \n",t1[i],yout[i][n2],yout[i][n3]);
                  break;
                case 3:
                  printf("  %15.5f    %15.5f    %15.5f    %15.5f    \n",t1[i],yout[i][n2],
                                yout[i][n3],yout[i][n4]);
                  break;
                default:
                  printf(" cannot output more than 3 varibles.\n");
            }
        }
    }
/* ------------------ CSS02.C 结束---------------------- */
```

实验三　随机数产生及均匀性与独立性测试

一、实验目的

(1) 学习几种产生随机数的方法。

(2) 掌握随机数均匀性及独立性的方法。

二、实验内容

(1) 利用 DESS01.C 程序对如下 PMMLCG 进行 χ^2 均匀性检验及独立性检验($\alpha=0.05$)

$$Z_i = 16\,807Z_{i-1}(\mathrm{mod}2^{31}-1)$$

样本点数　$n=10\,000$　　$K=10$　(均匀性)

　　$N=1\,000$　　$j=1$　　(独立性)

(2) 修改 DESS01.C 中的随机数发生器程序,实现对如下组合发生器进行 χ^2 均匀性检验及独立性检验($\alpha=0.05$)

$$Z_i = 25\,173Z_{i-1}+13\,849 \quad (\mathrm{mod}\ \ 65\,536) \quad Z_0=1.0$$

$$Z_i^2 = Z_{i-1}^2+Z_{i-2}^2 \quad (\mathrm{mod}\ \ 128)Z_0^2=1.0 \quad Z_1^2=1.0$$

(说明:首先产生 l 个 Z' 得到 l 个 U',形成数组 $V=(U_1,U_2,\cdots,U_l)$,l 可取为 128,然后产生在[1,l]上分布的随机整数(由 Z^2 产生)I。返回 U_I 作为组合发生器的第一个随机数,接着由 Z 产生一个随机数取代 U_I,再由 Z^2 产生新的 I,依次下去。)

要求:样本点数　$n=5\,000$　　$K=10$　(均匀性)

　　$N=1\,000$　　$j=1$　　(独立性)

三、预习要求

(1) 阅读 DESS01.C 仿真程序;

(2) 按实验要求,编制有关子程序;

(3) 实验步骤说明。

四、实验报告

(1) 预习报告；

(2) 实验程序清单；

(3) 实验结果数据；

(4) 实验结果分析；

(5) 总结收获及体会。

五、DESS01.C 源程序清单

```
/*----------本程序用于测试随机数发生器的均匀性与独立性----------*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int SUB1();
double SUB2();
int nvalue;
double probd[25];
long int a1, c1, m1, x1;
void main()
{
    int i, m;
    float mn;
    int ntypes, nx[20], number, number1;
    long int z0, a0, c0, m0, x0;
    double x, xx1, y, yy1, yy2, ro1, ro2;
    double xka2, squ;
/*----------以下是测试均匀性----------*/
    printf("\n Following  is to test the uniform.");
    printf("\n Input the number of area:  ");
    scanf("%d", &ntypes);
    printf("\n Input the number of random variable:  ");
    scanf("%d", &number);
    printf("\n a,c,m is the parament of the random generator; ");
    printf("\n z0 is the seed of this random generator.");
    printf("\n Input a,c,m,z0=");
    scanf("%ld,%ld,%ld,%ld", &a1, &c1, &m1, &x1);
    a0 = a1;
    c0 = c1;
    m0 = m1;
    x0 = x1;
    z0 = x1;
    x = 1.0 / ntypes;
    probd[1] = x;
    for(i = 2; i <= ntypes; i++)
        probd[i] = x + probd[i-1];
    for(i = 1; i <= ntypes; i++)
        nx[i] = 0;
    nvalue = ntypes;
    for(i = 1; i <= number; i++)
```

```
{
    m = SUB1();
    nx[m] = nx[m] + 1;
}
mn = (float)number / (float)ntypes;
xka2 = 0.0;
for(i = 1; i <= ntypes; i++)
{
    squ = nx[i] - mn;
    xka2 += squ * squ / mn;
}
printf("\n a=%f    ,c=%f    ,m=%d    ,z0=%f    ,");
printf("\n The nuber of random=%d    ", a1, c1, m1, z0, number);
for(i = 1; i <= ntypes; i++)
    printf("\n Nmuber(%d)=%d", i, nx[i]);
printf("\n CHI-SQUARE=%f", xka2);
/*----------以下是测试独立性----------*/
printf("\n Following  is to test the independence!");
printf("\n Input the number of random variable!");
scanf("%d", &number1);
a1 = a0;
c1 = c0;
m1 = m0;
x1 = x0;
z0 = x1;
x = SUB2();
yy1 = 0.0;
yy2 = x;
xx1 = (x - 0.5) * (x - 0.5);
for(i = 2; i <= number1; i++)
{
    y = SUB2();
    xx1 += (y - 0.5) * (y - 0.5);
    yy1 += x * y;
    yy2 += y;
    x = y;
}
yy1 /= (number1 - 1.0);
yy2 /= number1;
yy2 *= yy2;
xx1 /= (number - 1.0);
ro1 = (yy1 - yy2) / xx1;
ro2 = fabs(ro1) * sqrt(number - 1.0);
printf("\n Ro=%f    , u=%f    ", ro1, ro2);
printf("\n ---------- Simulation Results----------");
printf("\n\n a=%f    ,c=%f    ,m=%d    ,z0=%f", a1, c1, m1, z0);
printf("\n ---------- The Uniform ----------");
for(i = 1; i <= ntypes; i++)
    printf("\n Nunber(%d)=&d", i, nx[i]);
printf("\n Total_number=%d,    CHI_SQYARE=%f", number, xka2);
printf("\n ---------- The Independence ----------");
```

```
    printf("\n Number of samples=%d      RO=%f      u=%f", number1, ro1, ro2);
}
int SUB1()
{
    double u;
    int i;
    u = SUB2();
    for(i = 1; i < nvalue; i++)
        if( u < probd[i])      return(i);
    return(nvalue);
}
double SUB2()
{
  x1 = (a1 * x1 + c1) % m1;
  return( (double)x1 / (double)m1);
}
/*----------DESS01 结束----------*/
```

实验四　仿真输出数据分析

一、实验目的

(1) 加深理解离散事件系统仿真输出数据分析的重要性；

(2) 学习使用批均值法进行输出数据分析；

(3) 学习使用重复删除法进行输出数据分析。

二、实验内容

有一加工中心对五类零件进行加工。待加工零件发送模型是：到达时间间隔服从指数分布，均值 0.5h(上限 1.0h，下限 0.25h)，按均匀分布确定零件类型。零件到达后按 FIFO 方式在加工中心前排成单一队列。每类零件的加工时间服从 Erlang 分布，参数如下(单位：h)：

第 1 类 Erlang(2，0.4)，上限 0.7，下限 0.2

第 2 类 Erlang(3，0.5)，上限 0.8，下限 0.3

第 3 类 Erlang(1，0.1)，上限 0.6，下限 0.05

第 4 类 Erlang(2，0.4)，上限 0.6，下限 0.3

第 5 类 Erlang(4，0.6)，上限 1.0，下限 0.2

仿真要求：

(1) 加工零件 1 000 个，统计计算队列平均长度及每个零件平均等待时间；

(2) 加工零件 1 000 个，按 10 批，每批长度为 100 分别进行统计每批的队列长度及每个零件平均等待时间；

(3) 加工零件 1 000 个，按 10 批，每批长度为 100 分别进行统计，但随机地删除每批的前 $n(0<n<100)$ 个零件，计算平均队长及每个零件平均等待时间。

三、预习要求

(1) 阅读 DESS02.C 程序;

(2) 按实验要求修改 DESS02.C 有关模块;

(3) 实验步骤说明。

四、实验报告

(1) 预习报告;

(2) 实验用程序清单及输出结果清单;

(3) 估计批均值法仿真时的总平均队长及零件总平均等待时间及各自的置信区间(α=0.05);

(4) 估计重复删除法仿真的总平均队长及零件总平均等待时间及各自的置信区间(α=0.05);

(5) 比较三种不同统计分析方法的优缺点。

五、DESS02.C 源程序清单

```
/*----------仿真输出分析程序 DESS02----------*/
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
double tne[3];
unsigned long nevnts, next, numcus, status,niq;
double batime[101];
unsigned long number, batch, blen, arrbal, depbal;
double aniq, time, tlevnt, totdel, marrvt, mservt;
double delavg[101],niqavg[101],tarrvl[101];
unsigned long z;
void init_vari();
void timing();
void arrive();
void depart();
double expon( double );
double drand();
unsigned long mod1(double,unsigned long);
void report();
void main()
{
    unsigned long nzero;
    nevnts = 2;
    printf("\n Input mean interarrival time of arrive:");
    scanf("%lf", &marrvt);
    printf("\n Input mean service time:");
    scanf("%lf", &mservt);
    printf("\n Input Batchs:");
    scanf("%ld", &batch);
    printf("\n Input lengths of Batch:");
    scanf("%ld", &blen);
    init_vari();
```

```
        while( (arrbal < batch) || (depbal < batch) )
        {
                timing();
                if(next == 1)
                {
                    number++;
                    nzero = mod1((double)number,blen);
                    if(nzero == 0) printf("\n Arrival time= %lf     Number= %ld", time, number);
                }
                if(next ! = 2) arrive();
                else depart();
        }
        report();
    }
    void init_vari()
    {
        int i;
        z=1;
        time = 0.0;
        status = 0;
        niq = 0;
        tlevnt = 0.0;
        number = 0;
        numcus = 0;
        totdel = 0.0;
        aniq = 0.0;
        for(i = 1; i <= 100; i++)
        {
            delavg[i] = 0.0;
            niqavg[i] = 0.0;
            batime[i] = 0.0;
        }
        arrbal = 0;
        depbal = 0;
        tne[2] = 1.0E+30;
        tne[1] = time + expon(marrvt);
    }
    void timing()
    {
        unsigned long i;
        double rmin;
        rmin=1.0E+29;
        next=0;
        for(i = 1; i <= nevnts; i++)
        {
            if(tne[i] < rmin)
            {
                rmin = tne[i];
                next = i;
            }
        }
```

```
    if(next == 0)
    {
        printf("\n Event list empty!");
        exit(1);
    }
    time = tne[next];
}
void arrive()
{
    unsigned long num1, num;
    double delay;
    tne[1] = time + expon(marrvt);
    if(status ==1)
    {
        aniq += niq * (time - tlevnt);
        tlevnt = time;
        num1 = mod1((double)number,blen);
        if(num1 == 0)
        {
          arrbal++;
          batime[arrbal]=time;
          niqavg[arrbal] = aniq / (time-batime[arrbal-1]);
          aniq = 0.0;
        }
        niq++;
        if(niq > 100)
        {
            printf("\n Overflow of the array tarrvl!");
            exit(1);
        }
        tarrvl[niq] = time;
    }
    else
    {
        numcus++;
        num1 = mod1((double)number,blen);
        if(num1 == 0)
        {
          arrbal++;
          batime[arrbal]=time;
          niqavg[arrbal] = aniq / (time-batime[arrbal-1]);
          aniq = 0.0;
          }
          num = mod1((double)numcus, blen);
          if(num == 0)
          {
              depbal++;
              delavg[depbal] = totdel / blen;
              totdel = 0.0;
          }
          status = 1;
```

```
            tne[2] = time + expon(mservt);
        }
    }
    void depart()
    {
        unsigned long num, i;
        double delay;
        if(niq == 0)
        {
            status = 0;
            tne[2] = 1.0E+30;
            return;
        }
        aniq += niq * (time - tlevnt);
        tlevnt = time;
        niq--;
        delay = time - tarrvl[1];
        totdel += delay;
        numcus++;
        num = mod1((double)numcus,blen);
        if(num == 0)
        {
            depbal++;
            delavg[depbal] = totdel / blen;
            totdel = 0.0;
        }
        tne[2] = time + expon(mservt);
        for(i = 1; i <= niq; i++)
                tarrvl[i] = tarrvl[i+1];
    }
    void report()
    {
        unsigned long numtot, i;
        double sum1, sum2;
        double avgdel, avgniq;
        printf("\n ========== Single_Server Queueing System ==========");
        printf("\n\n -------- Parameter of the system model --------");
        printf("\n Mean Interarrival time= %lf      minutes.", marrvt);
        printf("\n Mean service time= %lf      minutes.", mservt);
        numtot = blen * batch;
        printf("\n Number of batches= %ld      ", batch);
        printf("\n Length of a batch= %ld      ", blen);
        printf("\n Number of customer= %ld      ", numtot);
        printf("\n\nBatch        Average delay      Average length");
        for(i = 1; i <= batch; i++)
        {
            printf("\n%3ld %16.5lf %18.5lf", i, delavg[i], niqavg[i]);
            avgdel += delavg[i];
            avgniq += niqavg[i];
        }
        avgdel /= batch;
```

```
    avgniq /= batch;
    printf("\n\n------ Results of simulation -------");
    printf("\n Average delay in queue: %10.3lf minutes",avgdel);
    printf("\n Average number in queue: %10.3lf",avgniq);
    sum1 = 0.0;
    sum2 = 0.0;
    for(i = 1; i <= batch; i++)
    {
        delavg[i] = avgdel - delavg[i];
        niqavg[i] = avgniq - niqavg[i];
        delavg[i] *= delavg[i];
        niqavg[i] *= niqavg[i];
        sum1 += delavg[i];
        sum2 += niqavg[i];
    }
    sum1 /= (batch-1);
    sum2 /= (batch-1);
    printf("\n Varience of average dealy in queue: %15.5f", sum1);
    printf("\n Varience of number in queue: %15.5f", sum2);
}
double expon( double rmean )
{
    double u;
    u = drand();
    return( -rmean * log(u));
}
double drand()
{
    double x;
    x = 65539.0 * z + 17325151.0;
    z = mod1(x, 2147483638);
    return ( z / 2147483638.0 );
}
unsigned long mod1(double dd1,unsigned long dd2)
{
    unsigned long temp,temp1;
    temp1 = dd1/dd2;
    temp = dd1-temp1 * dd2;
    return(temp);
}
```

附录 2
英文缩写词索引

A

ACSL (Advanced Continuous Simulation Language)先进的连续仿真语言,第 5 章
ADAMS(Automatic Dynamic Analysis of Mechanical System)机械系统动力学仿真分析软件,Mechanical Dynamics Inc. 产品,第 2 章
ADS (Advanced Distributive Simulation)先进分布仿真,第 1 章
AHP (Analytic Hierarchy Process)层次分析法,第 1 章
ALSP (Aggregate Level Simulation Protocol)聚合级仿真协议,第 1 章
AMRF(Advanced Manufacturing Research Facility)先进制造研究设施,第 1 章
ARPA(Advanced Research Project Agency)美国国防部高级研究计划署,第 12 章
API(Application Program Interface)应用程序接口,第 5 章
ANSYS 美国有限元分析软件公司的名称,也是其产品的名称,第 2 章
AS(Activity Scanning)活动扫描法,第 9 章
ASTRAL (Analog Schematic Translator to Algebra Language) 代数语言模拟示意翻译器,第 5 章
AVM(Antithetic Variates Method)对偶变量法,第 10 章

B

BDF(Back Difference Formula)后向差分公式,第 3 章
BLODC(Block Diagram Compiler)框图编译器,贝尔实验室开发,第 5 章
BOM(Based Object Model)基于对象的模型,第 1 章
BOOM(Binocular Omni-Oriented Monitor)双目潜望镜式的全景器,第 12 章

C

C4I(Command, Control, Communication, computer, Intelligence)军事术语,意为自动化指

挥系统,第1章
CAD(Computer Aided Design)计算机辅助设计,第1章
CC-NUMA(Cache-Coherent Non-Uniform Memory Architecture)一致性高速缓存非均匀存储访问体系结构,实际上是一个分布式共享存储处理机的多处理机系统,第1章
CCCS (Current Controlled Current Source)电流控制电流源,第2章
CCVS (Current Controlled Voltage Source)电流控制电压源,第2章
CEL(Current Events List)当前事件表,第9章
CGF(Computer Generated Force)计算机生成兵力,第11章
CIMS(Computer Integrated Manufacturing System)计算机集成制造系统,第1章
CORBA(Common Object Request Broker Architecture)公共对象请求代理结构,第11章
COT (Conditional Output Time)条件输出时间,第11章
CPU(Central Process Unit)中央处理单元,第1章
CRN(Common Random Numbers)公共随机数法,第10章
CRT(Cathode Ray Tube)阴极射线管,第12章
CSL(Control and Simulation Language)控制和仿真语言,第9章
CSMP(Continuous System Modeling Program)连续系统建模程序,第5章
CSS(Continuous System Simulation)连续系统仿真,第4章
CSSL(Continuous System Simulation Language)连续系统仿真语言,第5章
CVM (Control Variates Method)控制变量法,第10章

D

DADS(Dynamic Analysis and Design System)动力学分析与设计系统,CADSI公司产品,第2章
DARE (Differential Analyzer Replacement)微分分析器,第5章
DARPA(Defense Advanced Research Projects Agency)美国国防高级研究计划署,第1章
DAS(digital analog simulation)数字模拟仿真语言,第5章
DCOM(Distributed Component Object Model)分布式组件对象模型,第11章
DDM(Data Distribution Management)数据分发管理,第11章
DIS(distributed interactive system)分布交互式系统,第1章
DMSO(Defense Modeling and Simulation Office)美国国防建模与仿真办公室,第11章
DoD(Department of Defense)美国国防部,第11章
DR(Dead Reckoning)预推,第11章
DSL(Digital Simulation Language)数字仿真语言,第5章
DTW(Dynamic Time Warping)动态时间规整,第12章
DYNAMO系统动力学建模语言,第5章
DYSAC (Digitally Simulated Analog Computer)数字仿真式模拟计算机,第5章

E

ECSL(Extended Control and Simulation language)扩展的控制和仿真语言,第 9 章
ECOT (Earliest Conditional Output Time)最小的条件输出时间,第 11 章
EIT (Earliest Input Time)最小输入时间,第 11 章
EMC(Electro Magnetic Compatibility)电磁兼容性,第 1 章
EOT(Earliest Output Time)最小输出时间,第 11 章
ES(Event Scheduling)事件调度法,第 9 章

F

FCFS (First Come First Served)先到先服务,第 11 章
FEL(Future Events List)将来事件表,第 9 章
FEM(Finite Element Method)有限元方法,第 6 章
FIFO(First In First Out)先到先服务,第 7 章
FOM(Federation Object Model)HLA 联邦模型,第 1 章

G

GASP (General Activity Simulation Program)通用活动仿真程序,第 5 章
GIS (Geographic Information System)地理信息系统,第 1 章
GI/G/S(General Independent/ General/service)一般独立分布/一般分布/服务台数目,第 7 章
GPS(Global Positioning System)全球定位系统,第 11 章
GPSS(General Purpose Simulation System)通用仿真系统,第 9 章
GSL(General Simulation Language)通用仿真语言,第 5 章
GVT (Global Virtual Time)全局虚拟时钟,第 11 章

H

HCSE(Hierarchical Control System Emulator)递阶控制系统模仿器,第 1 章
HLA(High Level Architecture)高层体系结构,第 1 章
HMD(Head Mounted Display)头盔显示器,第 12 章
HPC(High Performance Computer)高性能计算机,第 1 章
HRTF(Head-Related Transfer Function)与头相关的传递函数,第 12 章
HTML(Hyper Text Markup Language)超文本标记语言,第 5 章

I

IBM(international business machine)国际商用机器公司,第 1 章
IBR(Image Based Rending)基于图像绘制,第 12 章
IEEE(International Electric/ Electrical Engineers)国际电子电气工程师协会,第 1 章
IGRIP 达索公式的仿真软件 Deneb 的一个模块,用于机器人仿真, 第 1 章

IID(Independent and Identically Distributed)独立同分布,第 10 章
I/O(input/output)输入输出,第 5 章
IPD(Interaural Pressure Difference)混响压力差,第 12 章
IST(Institute of Simulation & Training)美国 Central Florida 大学的仿真与训练研究所,第 11 章
ITD(Interaural Time Difference)混响时间差,第 12 章

K

KCL(Kirchhoff's Current Law)基尔霍夫电流定律,第 2 章
KVL(Kirchhoff's Voltage Law)基尔霍夫电压定律,第 2 章

L

LAN(Local Area Network)局域网,第 11 章
LCD(Liquid Crystal Display)液晶显示,第 12 章
LCG(Linear Congruential Generator)线性同余发生器,第 8 章
LIFO(Last In First Out)后到先服务,第 7 章
LMS(Load Management System)负载管理系统,第 11 章
LP(Logical Process)逻辑过程,第 11 章
LPC(Linear Prediction Compose)线性预测合成,第 12 章
LRC (Local RTI Component)本地 RTI 部件,第 11 章
LVT (Local Virtual Time)局部虚拟时间,第 11 章

M

MAS(Multi-Agent System)多智能体系统, 第 1 章
MATLAB(Matrix Laboratory)矩阵实验室,Math Works 公司产品,第 5 章
MDA (Model Driven Architecture)模型驱动体系结构,第 5 章
MIDAS (Modified Integration Digital Analog Simulator)改进型积分数字模拟仿真器,第 5 章
MIMD(Multiply Instruction Multiply Data)多指令多数据流,第 11 章
MISD(Multiply Instruction Single Data)多指令单数据流,第 11 章
MPP(Massively Parallel Processing)大规模并行处理, 第 11 章
M&S(Modeling & Simulation)建模与仿真, 第 1 章

N

NASA(National Aeronautics and Space Administration) 美国国家航空航天局,第 12 章
NEESGrid(Network for Earthquake Engineering Simulation grid)美国地震工程仿真网络网格,第 11 章

O

OGSA(Open Grid Service Architecture)开放网格服务框架,第 11 章
OMDT(Object Model Development Tool)对象模型开发工具,第 11 章
OMT(Object Model Template)对象模型模板,第 11 章
OpenSML (Open Simulation Modeling Language)开放的仿真建模语言,第 5 章

P

Pamcrash 碰撞冲击仿真软件,ESI 集团(ESI GROUP)产品,第 2 章
PARTNER (Proof of Analog Results Through Numerical Equivalent Results)数字等价模拟求证,第 5 章
PDE(Partial Differential Equation)偏微分方程,第 6 章
PDU(Protocol Data Units)协议数据单元,第 11 章
PI(Process Interactive)进程交互法,第 9 章
PMMLCG (Prime Modulus Multiplicative LCG)素数取模乘同余法,第 8 章

Q

QUEST(Queueing Event Simulation Tool)事件排队仿真工具,Delmia 公司的产品,第 9 章

R

RKM-45(Runge-Kutta-Merson)四阶龙格-库塔-默森法,第 3 章
RTI(Run-Time Infrastructure)HLA 运行支撑环境,第 1 章

S

SEDRIS(Synthetic Environment Data Representation and Interchange Specification)综合环境数据表示与交换标准,第 1 章
SF Express 美国国防部下属的国防先进研究计划处资助、由加利福尼亚理工学院负责完成的,其目标是模拟尽可能多的战斗单位,第 11 章
SIMAN(SIMulation ANalysis)仿真语言,C. Dennis Pegden 公司产品,第 9 章
SIMNET(SIMulator NETworking)美国 1998 年实施的分布式仿真计划,第 1 章
SIMPACK 多体动力学分析软件,德国航天局产品,第 2 章
SISD(Single Instruction Single Data)单指令单数据流,第 11 章
SIMD(Single Instruction Multiply Data)单指令多数据流,第 11 章
SLAM-Ⅱ,一种仿真语言,Pritsker 和 Associates 公司产品,第 9 章
SMP(Symmetrical Multi-Processing)对称多处理机,第 1 章
SOA(Service Oriented Architecture)面向服务框架,第 11 章
SOAP(Simple Object Access Protocol)简单对象访问协议,第 11 章
SOM(Simulation Object Model)HLA 仿真对象模型,第 1 章
SPEEDS(SPEculative and Exploratory Design in Systems Engineering)嵌入式系统开发中

的欧盟第六框架项目，第 11 章
SPL(Simulation Programming Language)仿真编程语言，第 9 章
SRML(Simulation Reference Markup Language)仿真参考标记语言，第 5 章
STRICOM(Simulation, Training and Instrumentation Command)美国陆军仿真训练装备司令部，第 11 章

T

TCP/IP(Transfer Control Protocol/Internet Protocol) 传输控制/网际协议，又叫网络通信协议，这个协议是 Internet 国际互联网络的基础，第 11 章
TP(Three Phases)三阶段法，第 9 章

U

UML(Unified Modeling Language)统一建模语言，第 5 章

V

VCVS(Voltage Controlled Voltage Source)电压控制电压源，第 2 章
VCCS (Voltage Controlled Current Source)电压控制电流源，第 2 章
VE(Virtual Environment)虚拟环境，第 12 章
VIEW(Virtual Interactive Environment Workstation)虚拟交换环境工作站，第 12 章
VLSI(Very Large Scale Integration)超大规模集成电路，第 11 章
VR(Virtual Reality)虚拟现实，第 11 章
VRML(Virtual Reality Modeling Language)虚拟现实建模语言，第 12 章

W

WAN(Wide Area Network)广域网，第 11 章

X

XML(eXtensible Markup Language)可扩展标记语言，第 1 章
XMSF(eXtensible Modeling and Simulation Framework)可扩展的建模与仿真框架，第 1 章